UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

VICTOR MARQUES MIRANDA

FERRAMENTAS DE AUXÍLIO AO DESENVOLVIMENTO DE PREDITORES E CONTROLADORES NEURAIS COMPLETOS A HORIZONTE FIXO PARA ROBÔS MÓVEIS MULTIARTICULADOS

VITÓRIA
2011

VICTOR MARQUES MIRANDA

FERRAMENTAS DE AUXÍLIO AO DESENVOLVIMENTO DE PREDITORES E CONTROLADORES NEURAIS COMPLETOS A HORIZONTE FIXO PARA ROBÔS MÓVEIS MULTIARTICULADOS

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Engenharia Elétrica.
Orientador: Prof. Dr. Edson de Paula Ferreira.
Co-orientador: Prof. Dr. Evandro Ottoni Teatini Salles.

VITÓRIA
2011

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Central da Universidade Federal do Espírito Santo, ES, Brasil)

Miranda, Victor Marques, 1985-
M672f Ferramentas de auxílio ao desenvolvimento de preditores e controladores neurais completos a horizonte fixo para robôs móveis multiarticulados / Victor Marques Miranda. – 2011.
168 f. : il.

Orientador: Edson de Paula Ferreira.
Coorientador: Evandro Ottoni Teatini Salles.
Dissertação (Mestrado em Engenharia Elétrica) – Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Robótica. 2. Robôs móveis. 3. Robôs - Sistemas de controle. 4. Sistemas de controle inteligente. 5. Redes neurais (computação). 6. Interface gráfica com o usuário (sistemas de computação). I. Ferreira, Edson de Paula. II. Salles, Evandro Ottoni Teatini. III. Universidade Federal do Espírito Santo. Centro Tecnológico. IV. Título.

CDU: 621.3

VICTOR MARQUES MIRANDA

FERRAMENTAS DE AUXÍLIO AO DESENVOLVIMENTO DE PREDITORES E CONTROLADORES NEURAIS COMPLETOS A HORIZONTE FIXO PARA ROBÔS MÓVEIS MULTIARTICULADOS

Dissertação submetida ao programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para a obtenção do Grau de Mestre em Engenharia Elétrica.

Aprovada em 25 de Novembro de 2011.

COMISSÃO EXAMINADORA

Prof. Dr. Edson de Paula Ferreira - Orientador
Universidade Federal do Espírito Santo

Prof. Dr. Evandro Ottoni Teatini Salles - Co-orientador
Universidade Federal do Espírito Santo

Prof. Dr. Klaus Fabian Côco
Universidade Federal do Espírito Santo

Prof. Dr. José Geraldo das Neves Orlandi
Instituto Federal do Espírito Santo

“Tudo posso Naquele que me fortalece”.

(*Filipenses 4:13*)

Aos meus pais, Benone e Marcia; a minha avó, Ledir; a minha namorada, Walesca; e a toda minha família.

Agradecimentos

Primeiramente agradeço a Deus por mais essa conquista e por sempre iluminar e abençoar o meu caminho e ser presença constante em minha vida.

Agradeço, em especial, a meus pais, Marcia e Benone, e a minha avó Ledir, pessoas que amo e admiro, sempre presentes e que são exemplos e referências para mim; a meus primos, Jackeline, Vander e Lucas, os quais considero irmãos e pais, e que me proporcionaram uma das maiores alegrias por mim já vivenciadas; e a toda minha família. Pela educação, amor, amizade, confiança e incentivo. Agradeço, em especial, a minha namorada, Walesca, pelo amor, carinho, apoio e compreensão. Por todos os momentos que passamos juntos nessa caminhada e por suas virtudes, as quais a torna especial, essencial e única. Enfim, agradeço a todas essas pessoas por suas capacidades de indicar direções, de retirar os excessos do dia-a-dia que nos tornam pesados e por serem pontes que nos fazem chegar aos lugares mais distantes de nós mesmos, despertando o que temos de melhor.

Agradeço a todos os colegas do LAI 2, pelo apoio, amizade e ambiente harmônico e respeitoso de trabalho; a UFES que nos proporcionou um ambiente adequado para a pesquisa e para a realização deste trabalho; e a CAPES pelo apoio financeiro concedido.

Agradeço ao meu não só professor orientador, mas também amigo e parceiro, Edinho. Pela atenção e presença constante, pela paciência, pelo conhecimento proporcionado e pelo tempo destinado aos seus alunos. Pela confiança que sempre teve em mim, principalmente, durante o projeto, no auxílio aos seus outros alunos e durante a realização dos artigos, obras essas de nosso sucesso. Por todos os momentos, desde as dificuldades às conquistas, os quais foram importantes para meu amadurecimento acadêmico.

Agradeço ao meu co-orientador Evandro; volto a reforçar: uma pessoa com a qual sempre quis trabalhar. Pela atenção prestada, pelos conselhos e direções em busca de soluções, pela amizade e confiança.

Por fim, agradeço aos demais membros da banca examinadora, composta pelos professores Klaus Fabian e José Geraldo Orlandi, dizendo ser uma satisfação em ser avaliado pelos mesmos.

Resumo

A navegação de robôs é um tema que tem motivado vários trabalhos acadêmicos e aplicados. Dentre os diferentes tipos de robôs móveis, os multiarticulados constituem objeto deste trabalho, onde são apresentadas uma nova sistemática e novas ferramentas para o desenvolvimento de preditores e controladores completos, com horizonte temporal fixo, utilizando redes neurais *feedforward* estáticas na descrição dos movimentos à ré de robôs móveis multiarticulados, no espaço de configuração. Os preditores são necessários para operação assistida ou para servir de núcleo em simuladores para análise de estratégias de navegação e para a síntese e validação de controladores. A sistemática proposta e as ferramentas desenvolvidas são gerais. A ferramenta criada visa facilitar os processos de criação, treinamento e validação de diversas topologias de redes neurais e de atribuição de seus parâmetros para o treinamento. Além disso, facilita a determinação do horizonte de predição, considerando o fato de essa ser uma tarefa exaustiva. Inicialmente, se dá o estudo de um protótipo, a ser utilizado na coleta de dados e na síntese e validação das redes, de grande similaridade a um veículo real, guardadas as devidas proporções, e de sua cadeia cinemática. Numa segunda etapa, é efetuada a modelagem do robô por meio de redes estáticas com variados horizontes de predição, sendo planejada, para isso, uma estratégia destinada à coleta de dados para posterior treinamento dos modelos. São necessárias, durante essa etapa, algumas manipulações sobre os dados que permitam colocá-los numa forma apresentável às redes. Além de dados reais, o conjunto de treinamento é composto de dados gerados a partir do modelo das condições singulares de giro. São apresentados modelos para os ângulos de giro e para os ângulos críticos, deduzidos a partir de um modelo analítico complexo original das equações gerais de movimento à ré de um robô multiarticulado, no espaço de configuração. Esse mesmo modelo é utilizado para gerar preditores analíticos a serem também usados em conjunto com controladores neurais. O uso de modelo para as configurações singulares é justificado porque as condições de giro são situações de equilíbrio instável, uma vez que, em malha aberta, não permitem a obtenção prática de dados. O modelo para os ângulos críticos define os limites, antes do *jackknife*, das variáveis de configuração. Finalmente, é apresentada a obtenção de controladores inversos, diretamente a partir dos dados ou indiretamente a partir do preditor, no espaço de configuração. A validação do controlador é, por fim, realizada para revelar a capacidade do mesmo em seguir as referências dentro de um horizonte e de um erro aceitáveis, sem que ocorram situações críticas e de *jackknife*, e de forma a manter a convexidade ou o giro.

Abstract

The robotic navigation is a subject that has motivated several academic and practical works. Among all the different kinds of mobile robots, the multi-articulated one is the focus of this work. This dissertation presents new systematic and new tools for the development of full neural predictors and controllers, with fixed time horizon, based on static multilayer feedforward networks, when describing backward movements of multi-articulated mobile robots, in the configuration space. The predictors are necessary for robot's assisted tasks and useful to be used as cores in simulators to analyze navigation strategies and for controller's synthesis and validation. The proposed systematic and the developed tools are general. The implemented tool helps to make easier the processes of creating, training and validating several topologies of neural networks and setting their training parameters. Besides, this tool helps to find a prediction horizon, for a given robot, regarding the fact that it is an exhaustive task. At first, the characteristics of the prototype used in this work, similar to a real vehicle, keeping the proper scales, and its kinematic chain are presented. Then, the robot is modeled using static neural networks with different prediction horizons. It is established a strategy for the data acquisition in order to obtain a representative database that will be used for training and validation of the models. It is implemented a preprocessing step, where some manipulations are done on data until they can reach an appropriate format for predictors training. The training data set is composed by real data acquired from measurements of the prototype and by data generated from circular singular conditions models. This work presents models for the singularities and for the critical angles extracted from an original analytical model of general backward movement equations of a multi-articulated mobile robot, in the configuration space. The same model is used to generate analytical predictors, which are used with the neural controllers. The use of models for singularities is necessary because the singular conditions are situations of unstable equilibrium, which makes impossible to obtain enough data from open loop real systems. The model for critical angles defines the range of the configuration variables before the jackknife. Finally, it is shown the generation of inverse controllers directly from the collected data and from the singular models or indirectly from the predictor, in the configuration space. The validation of the controller aims to reveal its capacity of following the references within an acceptable horizon and error, avoiding jackknife situations and keeping the convexity and the circular movement.

Sumário

Lista de Figuras

Lista de Tabelas

Nomenclatura

Símbolos, Siglas e Unidades

Símbolo	Descrição
RMMA	Robô Móvel Multiarticulado
AGV	*Automatic Guided Vehicle* (Veículo Guiado Automaticamente)
CVC	Composição de Veículos de Carga
CpPI	Controle por Propagação de Inferência
LAI	Laboratório de Automação Inteligente
PWM	*Pulse Width Modulation* (Modulação por Largura de Pulso)
RAM	*Random Access Memory* (Memória de Acesso Aleatório)
EEPROM	*Electrically-Erasable Programmable Read-Only Memory*
RF	Radiofrequência
CMOS	*Complementary Metal-Oxide-Semiconductor*
C.I.	Circuito Integrado
ASCII	*American Standard Code for Information Interchange*
DC	*Direct Current (Motor)*
E.S.C	*Electronic Speed Control*
A/D	*Analog-to-Digital Converter* (Conversor Analógico Digital)
MSE	*Mean Square Error* (Erro Médio Quadrático)
RMSE	*Root Mean Square Error*
DVD	*Digital Versatile Disc*
USB	*Universal Serial Bus*
S.I.	Sistema Internacional
A_1	Distância entre eixos do *truck*
A_i ($i > 1$)	Distância do ponto de engate dianteiro ao eixo
B_i	Distância do ponto de engate traseiro ao eixo
γ	Ângulo de orientação das rodas dianteiras
θ_i	Ângulos de configuração
$\delta\theta_i$	Variações dos ângulos de configuração
$\underline{\theta}$	Vetor de configuração
$\delta\underline{\theta}$	Vetor das variações angulares das configurações
m	Metros (distância)

ms	Milissegundos (tempo)
m/s	Metros por Segundo (velocidade)
bits	*Binary Digit* (dígito binário)
KB	*Kilo-Bytes* (velocidade de transmissão de dados)
Kbps	*Kilo-bits por segundo* (velocidade de transmissão de dados)
Mbps	*Mega-bits por segundo* (velocidade de transmissão de dados)
KΩ	*Kilo-Ohm* (resistência)
mAh	*Mili-Ampère-Hora* (carga elétrica)
volts	*Volts* (voltagem)
MHz	*Mega-Hertz* (frequência)
GHz	*Giga-Hertz* (frequência)
%	Porcentagem
°	Posição Angular (graus)

Capítulo 1: Introdução

A robótica representa um campo relativamente recente da tecnologia moderna que atravessa as fronteiras da engenharia. A compreensão da complexidade dos robôs e de suas aplicações requer conhecimentos de diversas áreas que vão desde a elétrica, a mecânica, a automação e a instrumentação à computação, à economia e à matemática.

A navegação de robôs, em particular, é um tema que tem motivado vários trabalhos acadêmicos. A literatura apresenta de modo conclusivo a importância da pesquisa em modelagem, controle e navegação de robôs móveis, especialmente de robôs móveis multiarticulados - RMMA's (Ferreira, 1999, 2004; Kulitz, 2003), os quais constituem o foco deste trabalho.

Um RMMA, similarmente a um veículo multiarticulado, é caracterizado por sua cadeia mecânica articulada, em geral, constituída de semi-reboques (ou *trailers*), responsáveis pelo transporte da carga, engatados a um veículo de tração[1] motorizado (ou *truck*). É interessante destacar que o estudo desse tipo de robô abrange conceitos da robótica móvel aliados aos dos robôs manipuladores.

Apesar de outros tipos de arquiteturas de robôs móveis serem tratadas na literatura, a cadeia mecânica em questão é a mais geral, pois possui uma gama de aplicações diversificada, que vai desde robôs para armazéns automáticos, dutos ou outros espaços restritos, pátios de carga e descarga, até grandes veículos multiarticulados para transporte de carga em rodovias, portos e aeroportos, os chamados CVC (Composição de Veículos de Carga). A Figura 1.1 ilustra diferentes tipos de veículos articulados.

[1] Equivalente, no Brasil, ao chamado cavalo-mecânico.


Shell
Shell

Figura 1.1 Aplicações de veículos multiarticulados (*multi-trailer-systems*).

1.1 Motivação

Progressos na navegação robótica poderiam contribuir para redução de custos e melhoria de eficiência no setor industrial e na cadeia logística brasileira, sendo essa última a mais ardorosa por inovações e otimizações de processos.

No caso da cadeia logística brasileira, o transporte rodoviário de cargas é, sem dúvida, um dos pontos de maior vulnerabilidade da mesma. Responsável pelo deslocamento de 63% das cargas transportadas pelo país (BgmRodotec, 2011), o setor enfrenta problemas tais como as longas jornadas dos motoristas e autônomos; a prática de sobrecarregar os veículos para tentar compensar os fretes raquíticos; o péssimo estado de uma parte importante da malha rodoviária (muitas vezes em decorrência do fator anteriormente citado); e a utilização de uma

frota obsoleta de caminhões, o que provoca, entre outros graves inconvenientes, insegurança nas operações, elevado custo de manutenção, poluição e consumo excessivos (Vianna, 2011).

Um fator que colaboraria diretamente na redução dos custos de transporte rodoviário de cargas e, por consequência, dos preços agregados ao consumidor final, seria a maximização da carga total transportada. Implementável de modo simples com o emprego de veículos multiarticulados, o que diminuiria a carga transportada por eixo, essa solução visa à otimização do uso da infra-estrutura de transportes, melhorando as condições de segurança, diminuindo a degradação das estradas e reduzindo o consumo de energia e a emissão de poluentes.

De modo similar, em atividades de manufatura, existe usualmente uma tarefa que move ou armazena a matéria-prima e o produto acabado. Essa atividade vem sendo progressivamente automatizada com o emprego dos veículos guiados automaticamente (AGV's), conforme exemplificados na Figura 1.2, que podem ser utilizados para movimentação de materiais, introduzindo sensíveis ganhos de tempo e redução de custos (Slack, 1997).

Figura 1.2 Exemplo de um AGV.

Os ganhos de custos, tempo e, consequentemente, produtividade proporcionados às células de manufatura e armazéns verticais em muito poderiam ser ampliados se, similarmente ao caso dos caminhões, os AGV's fossem dotados de uma maior capacidade de transporte.

Diante das vantagens e soluções apresentadas, há de se refletir sobre onde residem os impedimentos para a operacionalização de tais melhorias. A resposta a esta questão levando-

se em conta, dentre vários aspectos, principalmente as considerações técnicas, reside na complexa dirigibilidade de tais veículos. Apesar de serem largamente disseminados na malha rodoviária pela suas grandes capacidades de carga, os veículos multiarticulados apresentam baixa manobrabilidade, sobretudo em movimentos à ré.

1.2 Definição do Problema

A complexidade maior do controle de manobras de RMMA's está ligada aos movimentos em marcha à ré, posto que os para frente não introduzam dificuldades adicionais ao problema, o que também representa um dos maiores desafios da navegação autônoma de RMMA's. Uma tarefa cotidiana a muitos motoristas, como estacionar a composição[2] ou desviar de algum obstáculo, realizando uma manobra à ré aumenta consideravelmente seu grau de complexidade na medida em que o número de *trailers* acoplados ao *truck* aumenta, tornando-se, em vias gerais, praticamente impossível, em certas situações com mais de dois *trailers*, de ser efetuada única e exclusivamente por um operador humano. Realizar o movimento à ré em veículos articulados, por si só, exige do motorista uma razoável experiência, a qual se eleva com a ampliação do número de articulações.

O fato de o controle da composição ser todo realizado pelo cavalo mecânico torna o movimento à ré de veículos multiarticulados extremamente complexo, não-linear e de difícil modelagem (Kulitz, 2003; Pinheiro, 2004). Nessa situação, o sistema comporta-se como um pêndulo invertido múltiplo horizontal. Uma vez que os engates permitem a livre rotação, os ângulos de configuração[3] (em um ou mais engates da composição) podem aumentar, para qualquer valor do ângulo da direção, assumindo valores inadequados (iguais ou maiores aos seus respectivos valores máximos permissivos ou valores críticos), a partir dos quais quaisquer ações se tornam ineficazes em promover o controle do veículo. Essas situações constituem exemplos de configurações críticas e, uma vez atingidas, evoluem para o engavetamento ou *jackknife*[4] (dar um "L", em linguagem coloquial) com a continuidade do movimento para trás (Ferreira, 2004). Ao atingir o *jackknife*, a única ação possível consiste em movimentar o veículo para frente.

[2] Composição: conjunto formado pelo veículo de tração e semi-reboques.

[3] Os ângulos ou variáveis de configuração correspondem aos ângulos entre os sucessivos elementos da cadeia mecânica articulada (de forma sucinta, são os ângulos das articulações).

[4] É atribuído o nome *jackknife* por lembrar o movimento realizado por um canivete.

Nesse contexto, faz-se imprescindível desenvolver um sistema de supervisão e de controle para os movimentos à re de um RMMA, o que representaria uma importante ferramenta para auxiliar o motorista ou automatizar manobras e evitar configurações críticas da cadeia cinemática, evitando, assim, o *jackknife*.

1.3 Uma nova óptica de solução

A solução mais geral para o problema de controle de manobras e de navegação autônoma de RMMA's passa pela implementação de um sistema de supervisão e controle, como mostrado na Figura 1.3 e explicado a seguir.

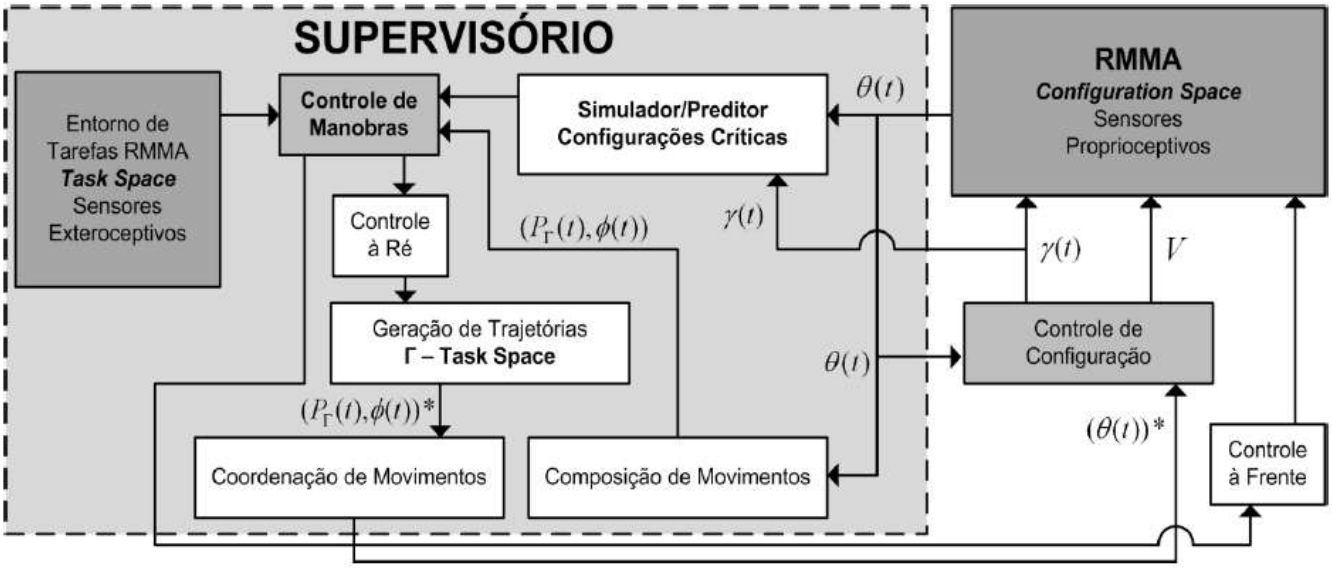


Figura 1.3 Esquemático do sistema de supervisão e controle de um RMMA.

Na automação flexível, visando uma maior generalidade das soluções, processos complexos ou serviços demandam que as ações de controle ou de predição sejam implementadas em nível de espaço de configuração de cada máquina ou subsistema, de modo desacoplado das ações definidas no espaço de tarefas (*task space*). Deste modo, no nível de configuração, os problemas de controle ou predição teriam referências, critérios e soluções caracterizados neste único espaço, evitando que o controlador opere implícitas inversões geométricas, o que pode conter singularidades. Essas premissas são válidas tanto para robôs manipuladores quanto para RMMA's.

Alguns autores propõem soluções diretamente no espaço de tarefas, porém carecem de generalidade. Por exemplo, Kinjo (2006) propõe uma solução neural diretamente no espaço

de tarefas, contudo ela é muito restrita, pois é desenvolvida para uma configuração com somente um *trailer* e a validação é apresentada somente em simulação. Assim, visando uma maior generalidade das soluções, o problema abordado no presente trabalho é formulado no espaço de configurações do robô.

Em navegação robótica, a caracterização das tarefas pode incluir restrições, obstáculos ou incertezas diversas e, assim, num nível de coordenação, estas tarefas seriam transformadas em ações de referência para um nível inferior de controle ou predição no espaço de configurações. Essa transposição entre os espaços de tarefa e de configuração é denominada Coordenação de Movimentos. O nível de coordenação seria implementado através de modelos inversos geométricos e/ou cinemáticos, obtidos na presença de restrições, segundo critérios pré-definidos.

A coordenação de movimentos é estabelecida, juntamente com outras ações necessárias, num sistema supervisório de manobras e navegação. Observa-se que no supervisório, a coordenação de movimentos via modelos é necessária para o caso de manobras automáticas ou assistidas, onde as referências no espaço de tarefas são caracterizadas por um operador humano. A idéia básica do emprego do sistema supervisório reside em mudar o espaço onde o operador executa sua tarefa, permitindo-lhe interagir com o processo de maneira mais simples e através de comandos menos especializados.

O sistema supervisório, portanto, pode ser construído segundo duas abordagens distintas: manobras assistidas e manobras automáticas. Na primeira delas, o sistema auxilia o usuário prevendo, por meio de simulações, o movimento para trás. Isto permite ao usuário saber qual será o comportamento dos ângulos das articulações do veículo, caso a composição se movimente para trás, a partir da configuração[5] atual e de certa orientação das rodas dianteiras do *truck*. Este sistema tem a funcionalidade de um preditor (Ferreira, 2004). Como exemplo prático, um preditor seria uma ferramenta, um *software* instalado no computador de bordo do veículo. Sob o ponto de vista teórico, um preditor é um modelo direto do robô articulado. Os preditores são necessários não só para operação assistida, mas também para servir de núcleo em simuladores na análise de estratégias de navegação e na síntese e validação de controladores.

Na outra abordagem, o sistema é responsável pela realização da manobra. Neste caso, o usuário informa a trajetória no espaço de tarefas e o sistema supervisório fornece para o controlador de configuração as referências adequadas neste nível. Vale ressaltar que

[5] Uma configuração é uma combinação dos ângulos das articulações (ou ângulos de configuração).

praticamente toda tecnologia desenvolvida para RMMA's pode ser adaptada para a navegação ou controle, na execução otimizada de manobras automáticas ou assistidas de veículos articulados.

Na Figura 1.3, é explicitado um esquema básico de funcionamento de um sistema supervisório. Percebe-se que além da coordenação de movimentos, esse deve implementar um sistema de previsão de configurações críticas, para economia de movimentos em manobras. De fato, uma configuração crítica não é somente aquela em que algum ângulo de configuração atinge o seu valor crítico, o que pode ser detectado no nível mais baixo de controle. É também aquela em que o movimento à ré para levar a cadeia articulada para uma configuração de giro, a partir da qual ela possa ser controlada, faz com que algum ângulo de configuração atinja o seu valor crítico.

Como apresentado em Ferreira (2004), é necessário não somente os modelos singulares de giro e os modelos dos ângulos críticos[6] para caracterizar uma configuração crítica, mas, principalmente, um bom simulador ou preditor. Um sistema só pode ser controlado a partir de uma configuração de giro e, no movimento efetuado para atingi-la, pode ocorrer uma configuração onde um dos ângulos atinge o seu valor crítico. Isso significa que alguma configuração intermediária, ou mesmo a de partida, já representava uma configuração crítica. Logo, ter um bom simulador num nível supervisório permite detectar as configurações críticas e, feito isso, em vez de continuar o movimento para trás, o RMMA é parado e movido para frente. Isso traz uma grande economia de movimentos desnecessários.

A Figura 1.4 mostra uma aplicação real de um sistema supervisório.

Figura 1.4 Sistema supervisório instalado na cabina de um veículo multiarticulado.

[6] Os modelos singulares de giro e os modelos dos ângulos críticos serão discutidos ao longo do Capítulo 4.

1.4 Escopo

Existem na literatura poucas propostas de técnicas de modelagem e controle do movimento à ré de RMMA's. A execução dos movimentos à ré, em manobras ou navegação de RMMA's tem o modelo analítico, em geral, obtido com muitas suposições simplificadoras, ainda assim, este é não-linear e muito complexo (Ferreira, 2004). Com isso, este modelo impõe sérias restrições ao uso de técnicas de síntese tradicionais como em Petrov (2010).

Logo, como o modelo cinemático da cadeia mecânica articulada é altamente complexo e fortemente não-linear, impossibilitando qualquer afirmativa, à priori, sobre a existência de uma expressão analítica para sua inversa, constata-se que as técnicas mais indicadas à solução do problema de controle estão na inteligência computacional, via aproximações numéricas *model free*, neurais (Ferreira, 1999; Silva, 2003; Kinjo, 2006; Demcenko, 2008) ou *fuzzy* (Ferreira 2002, 2004, 2010; Kulitz, 2003), principalmente se deslizamentos, folgas, saturações, elasticidades, ruídos e diversas outras perturbações forem considerados no problema. Na literatura, este tipo de abordagem é denominado "inteligente" por ser parte da inteligência artificial ou inteligência computacional.

Em Ferreira (2010), foi proposta uma sistemática para a descrição dos movimentos à ré de RMMA's via preditores *fuzzy* a horizonte fixo de tempo, não completos[7]. Nele, fica evidente que a generalização para preditores completos e com maior granularidade leva rapidamente a uma "explosão" do número de regras. Esse fato é o maior inconveniente para o uso da abordagem *fuzzy*. Resta, então, a possibilidade da geração automática das regras a partir dos dados, o que simplificaria muito o esforço da modelagem *fuzzy*, porém é mostrado, também, que a geração automática pelos métodos conhecidos falha em completude e consistência. Por isso é desenvolvida uma sistemática *ad-hoc* para a finalização adequada dos modelos *fuzzy*. Com o aumento do número de variáveis linguísticas ou do número de graus de liberdade[8], essa sistemática se revela muito trabalhosa, tornando atrativa, portanto, a solução neural. Assim, para manter a precisão e a generalidade, com o uso de modelos completos, a alternativa mais interessante seria a utilização da abordagem neural, objeto deste trabalho.

[7] O conceito de preditor completo a horizonte fixo de tempo será abordado mais à frente.

[8] Número de graus de liberdade = quantidade de variáveis de configuração + 1;

1.5 Trabalhos Relacionados

O controle de configuração é uma etapa hierarquicamente inferior e, portanto, sua solução tem prioridade em relação às demais etapas. Em busca de soluções para essa etapa, alguns trabalhos foram realizados pelo grupo de pesquisa de controle de RMMA's do LAI-DEE-UFES (Laboratório de Automação Inteligente do Departamento de Engenharia Elétrica da Universidade Federal do Espírito Santo), ao qual esse trabalho pertence.

Os mais relevantes frutos destes estudos são: uma tese de doutorado (Kulitz, 2003), quatro dissertações de mestrado (Silva, 2003; Pinheiro, 2004; Rampinelli, 2008; Barreto, 2010) e quatro monografias (Rampinelli, 2006; Miranda, 2009; Pandolfi, 2009; Bertolani, 2011), além de inúmeros artigos publicados em congressos nacionais e internacionais. Dentre esses artigos, alguns desenvolvidos a partir desse trabalho merecem destaque: (Miranda e Ferreira 2011a, b, c; Miranda, Ferreira e Villela, 2011a, b).

Os trabalhos desenvolvidos no LAI-DEE-UFES visam principalmente à modelagem e ao controle de configuração angular de RMMA's em movimentos à ré e usam, em sua maioria, algoritmos de controle inteligente. Porém, uma atenção tem sido dada também ao desenvolvimento de preditores e controladores analíticos, ao uso de técnicas de visão computacional e à implementação de técnicas de identificação de sistemas e de sintonia de controladores. A seguir, serão comentados alguns desses trabalhos.

Em sua dissertação, Pinheiro (2004) desenvolveu um modelo para veículos multiarticulados e propôs um preditor para auxiliar o motorista durante manobras, ambos usando a teoria *fuzzy*. Kulitz (2003), em seu doutorado, implementou um simulador a partir de uma modelagem analítica do sistema e propôs uma técnica generalizada para o controle de robôs multiarticulados com qualquer número de elementos, a qual ele denominou de CpPI - Controle por Propagação de Inferência. Entretanto, naquele momento ele não dispunha de um protótipo adequado para testes e, por esta razão, o controlador não foi testado em uma situação mais realista.

Seguindo com as pesquisas, em 2008, Rampinelli (2008) propôs uma nova técnica de realimentação, baseada em visão computacional, representando, assim, uma opção ao uso de sensores elétricos nas articulações. Com o propósito de validar sua técnica, implementou o controlador proposto por Kulitz (2003) e realizou experimentos utilizando o protótipo real. Os testes realizados mostraram que o controlador não se comportou muito bem quando a referência era não-nula, ou seja, quando se desejava que o ângulo entre os dois últimos

trailers fosse diferente de zero. Nesses casos, o controlador apresentava um grande *offset* em regime estacionário, e, em alguns casos, não conseguia realizar a manobra (a composição entrava em engavetamento). Como pôde ser observado na dissertação de Rampinelli (2008), esse comportamento insatisfatório deveu-se ao controlador e não à realimentação.

As pesquisas mais recentes realizadas na UFES, envolvendo técnicas de controle inteligente, foram os trabalhos de Barreto (2010), Pandolfi (2009) e Miranda (2009). O dois primeiros implementaram controladores *fuzzy* dotados ou não de conhecimentos especialistas (*ad-hoc*) do operador; Barreto (2010), especificamente, concentrou esforços no desenvolvimento de uma plataforma experimental para ensaios de manobras com RMMA's, a qual constitui o protótipo do robô utilizado por este trabalho. Já a monografia de Miranda (2009), objeto de inspiração para esse trabalho, fez uso da teoria de controle neural inverso via modelo de referência para síntese de um controlador neural e, posterior, validação no protótipo disponível; vale destacar que o trabalho de Miranda (2009) foi a primeira tentativa do uso da abordagem neural usando o protótipo em questão.

No que diz respeito ao emprego de redes neurais, dois outros trabalhos presentes na literatura foram importantes à navegação de um RMMA em manobras de estacionamento e servem de base para diversos projetos nessa área.

Uma alternativa de controle de movimentação à ré, descrita na literatura, foi baseada nos trabalhos de Nguyen e Widrow (1990). Ambos utilizaram com sucesso redes neurais para tratar o problema de estacionar um caminhão, composto por um *truck* e um *trailer*, em uma posição pré-estabelecida, efetuando movimento à ré. A solução por eles empregada consistia no treinamento de uma rede para controlar a direção das rodas dianteiras, de forma a conduzir o caminhão ao ponto desejado de estacionamento. Apesar de a estratégia empregada ter alcançado o objetivo desejado, foram exigidos muito esforço para o treinamento da rede neural, responsável pelo controle, e uma topologia com grande número de camadas e neurônios por camada. O processo de treinamento passava por estacionar o caminhão, gradativamente, posicionando-o mais longe do ponto de estacionamento, a cada novo treinamento. Assim, o controlador neural memorizava as trajetórias que levariam o caminhão ao seu objetivo, para uma dada posição inicial.

Na tentativa de melhorar a abordagem do problema anterior, Jenkins e Yuhas (1993) mantiveram a proposta original de Nguyen e Widrow (1990) de utilizar a localização absoluta do caminhão e sua orientação como base para o aprendizado da rede na tomada de decisões, e inseriram como nova componente a habilidade (*know-how*) do operador na definição da

estratégia de manobra, aliando desta maneira às redes neurais a lógica nebulosa na tomada de decisão e minimizando a ocorrência de situações de *jackknife*.

Apesar da grande contribuição dessas duas propostas, não se poderia desconsiderar o fato de ambas as soluções trabalharem sob a hipótese de ambientes de navegação simples, sem obstáculos e estáticos e demonstrarem baixa capacidade de generalização. Na solução proposta pelos autores, mudanças na posição de estacionamento (garagem), por exemplo, requereriam um novo treinamento da rede neural empregada.

Uma proposta mais geral seria a de subdividir o problema em dois níveis: um nível de supervisão da navegação, onde seriam implementadas as regras de navegação e tomadas as decisões necessárias, e um nível de controle do robô articulado, onde se implantariam as decisões definidas pelo nível de supervisão, de forma otimizada e orientada a eliminar a ocorrência de *jackknife*.

Por fim, outras abordagens de inspiração, utilizando redes neurais para o controle de configuração de um RMMA com dois ou três graus de liberdade, foram empregadas nas seguintes situações:

- Aproximação numérica via rede neural dinâmica com controle neural inverso para protótipos reais, com pequena restrição no ângulo de controle [-60°, +60°], e ajuste *ad-hoc* da realimentação de estados mais controle PI para garantir robustez em malha fechada (Ferreira, 1999);
- Demonstração de aplicabilidade em simulação, sem restrição no ângulo de controle [-90°, +90°], utilizando rede estática multicamada na aproximação dos movimentos e ajuste *ad-hoc* do controlador linear em malha fechada (Silva, 2003). Este trabalho foi a primeira tentativa de se conseguir uma solução alternativa, mais simples, ao controle neural inverso baseado em rede dinâmica, proposto por (Ferreira, 1999). Embora os resultados em simulação indicassem a viabilidade da abordagem, a falta de sistematização e as restrições impostas na formulação do problema não permitiram a generalização e a expectativa de desempenho satisfatório em casos reais ou em problemas mais complexos.

  A síntese apresentada por Silva (2003) foi feita exclusivamente em simulação, onde todos os movimentos para trás (inclusive os movimentos em giro) foram gerados via modelo, em condições ideais. Essa estratégia não apresentaria o mesmo desempenho na prática devido à inexistência de perturbações, ruídos ou outros fatores e porque a informação sobre a velocidade é importante para assegurar um bom desempenho.

1.6 Objetivos e Metodologia Empregada

O presente trabalho vem também resgatar a abordagem de Silva (2003), via rede estática, mas visando à discussão de metodologias necessárias a prover resultados mais conclusivos quanto ao desempenho e à generalidade. É necessário esclarecer que sendo o robô um sistema dinâmico, seria necessária, à priori, uma rede dinâmica (recorrente) para representar o seu movimento, como em Ferreira (1999), contudo esta alternativa é muito custosa, principalmente na aquisição de dados necessários ao treinamento, o que demanda um maior esforço de implementação. Apesar de seus consideráveis potenciais no tratamento de certos problemas, tais como modelagem de sistemas dinâmicos, aplicações de controle adaptativo e processamento de séries temporais, as redes recorrentes ainda apresentam certas dificuldades de treinamento (Ballini, 2001), posto que muitos algoritmos de aprendizagem conhecidos sejam complexos e lentos (Lee, 2000). Além disso, considerando que, para movimentos à ré em baixas velocidades, o sistema comporta-se de maneira quase estática, optou-se pelo emprego de redes estáticas *feedforward*[9], o que constitui uma alternativa mais simples ao uso de redes dinâmicas.

Essa alternativa consiste na representação do sistema entre dois instantes de tempo, ou num horizonte fixo H, múltiplo do tempo de amostragem T (H = hT, h = 1,2,...), considerando como entradas, no início do intervalo, o ângulo de direção (γ), o vetor de configuração ($\underline{\theta}$) e as variações da configuração ($\delta\underline{\theta}$). Esta abordagem é um tipo de modelagem de um sistema dinâmico, usando uma aproximação de horizonte fixo H e pode ser adotada tanto em aproximações via lógica *fuzzy* quanto naquelas via redes neurais. Um dos objetivos desse trabalho é mostrar que esta segunda abordagem, muito mais simples, é viável.

Caberia, logo, ao preditor, baseado na informação corrente da configuração, suas variações e do ângulo de direção, informar, num horizonte H finito de tempo (horizonte de predição), as correspondentes variações da configuração. Ressalta-se que essa abordagem proposta é válida tanto para preditores quanto para controladores, sendo estes últimos desenvolvidos para atingir uma dada referência num horizonte de controle H. Além disso, levar em conta na entrada de cada rede neural (seja como preditor ou como controlador) variações de componentes da configuração inclui implicitamente informação sobre a

[9] Essas redes são particularmente mais simples de se trabalhar, respondendo melhor e de maneira mais rápida a algoritmos de aprendizagem mais consolidados, como é o caso do *Backpropagation* e suas variações, descritos no Apêndice 1.

velocidade do *truck*, por isso o mapeamento que caracteriza cada rede será denominado "completo", contrapondo-se ao mapeamento puramente geométrico entre configurações. Os detalhes da aplicação dessa sistemática em casos reais de predição e controle neurais são explicitados no decorrer dos próximos capítulos.

Um dos fatores aliados à viabilidade de realização deste projeto é que toda a estrutura necessária para sua execução já é conhecida e está disponível no LAI-DEE-UFES. Desta maneira, o projeto não gera grandes custos nem sofre atrasos para a aquisição de equipamentos e/ou novos elementos de *hardware*. Basicamente, a estrutura física necessária é composta por um protótipo realista adaptado de um veículo multiarticulado (composto por um *truck* e dois *trailers*), um computador e um sistema de comunicação com o computador.

No que diz respeito à metodologia empregada neste trabalho, inicialmente é estudada e analisada a melhor forma de fracionamento do universo de possibilidades de configurações iniciais do veículo, respeitando as limitações angulares de cada articulação (valores críticos) a partir das quais ocorre *jackknife*, de modo a representar bem a dinâmica do sistema. Na solução do problema, a primeira tarefa é a exaustiva coleta de dados. Exaustiva, na medida em que seja necessária a obtenção de uma imensa massa de dados, e complexa, tendo em vista a necessidade de definição de uma estratégia adequada para a obtenção de tais dados, de modo a se gerar um conjunto representativo que modele suficientemente o comportamento do RMMA. Faz-se, então, necessário o uso de protótipo de um veículo multiarticulado real, que auxilie na coleta de dados necessários ao treinamento, síntese e validação de preditores e de controladores.

A tarefa retratada de coleta dará subsídio, posteriormente, à geração de uma base de dados crua que, após sofrer o tratamento adequado[10], será utilizada no treinamento das redes que modelam a dinâmica do RMMA em movimentos à ré. Uma vez terminada a coleta, para cada aplicação específica, é implementada uma etapa de pré-processamento, onde são feitas manipulações sobre os dados que permitam colocá-los numa forma adequada ao treinamento das redes. Por exemplo, inicialmente é feita a conformação e filtragem dos dados provenientes de sensores (câmeras, potenciômetros, etc...); em seguida, são eliminados dados inconsistentes ou que ultrapassem os valores críticos; entre outros. Pode-se, também, eliminar os dados de configurações côncavas que estejam fora dos limites de treinamento desejado.

[10] Um exemplo prático de implementação da etapa de pré-processamento é apresentado no Capítulo 5, na seção 5.2.2.

Da massa de dados pré-processada, são extraídos dados correspondentes a cada rede (preditor ou controlador) com horizonte de predição característico, de modo a gerar novas bases representativas a cada uma[11], conforme os mapeamentos que as definem.

Neste trabalho, são apresentados modelos para as condições singulares de giro e para os ângulos críticos, extraídos de um modelo analítico original das equações de movimento à ré de RMMA's, no espaço de configurações, desenvolvido por Ferreira (2011), cuja sistemática de obtenção é detalhada em Pandolfi (2011). O modelo para os ângulos críticos é usado para definir os limites, antes do *jackknife*, das respectivas variáveis de configuração e é importante desde a coleta de dados ao treinamento e à validação das redes neurais. O uso de modelo para as singularidades é necessário porque as condições de giro são situações de equilíbrio instável, o que torna difícil ou impossível obter dados suficientes de sistemas reais em malha aberta. Vale destacar que além de dados reais, o conjunto de dados gerados a partir do modelo das condições singulares de giro irá compor as massas de dados para treinamento de preditores e controladores com horizontes de predição variados.

O processo de treinamento, por sua vez, exigirá o estudo de um algoritmo de aprendizado neural supervisionado adaptativo, sendo o *Backpropagation* o mais adequado para isso. A fase de treinamento é seguida pela fase de teste, onde um conjunto de dados não previamente utilizado no treinamento de uma determinada rede é submetido à sua entrada a fim de avaliar o seu desempenho ou sua capacidade de generalização. É desejável que uma rede treinada estime, reconheça e classifique dados desconhecidos ou incompletos, inferindo soluções similares aos padrões aprendidos, por meio de um processo de interpolação. Por fim, a validação de um preditor revela a habilidade do mesmo em seguir trajetórias, dentro das restrições e premissas em que foi treinado; e a validação do controlador verifica a capacidade do mesmo em controlar a planta, seguindo, de modo adequado, dentro de uma margem tolerável de erro, as referências impostas ao sistema evitando, assim, situações críticas e de *jackknife* durante as manobras efetuadas.

O corrente trabalho explicita duas formas de obtenção dos controladores neurais: diretamente a partir dos dados coletados e a partir daqueles gerados pelo modelo de singularidades, ou indiretamente a partir da planta, e em ambos os casos atuando como uma inversa aproximada (à esquerda) do preditor de horizonte H. A idéia da inversa consiste em cancelar os efeitos não-lineares da dinâmica da planta, gerando um sistema linearizado dentro de certo horizonte válido de atuação do controlador.

[11] Um exemplo prático de constituição dessas massas de dados, para cada horizonte, é apresentado no Capítulo 5, na seção 5.2.3.

Pode-se dizer que a ação do controlador, por sua vez, consiste no cálculo de um ângulo adequado para as rodas dianteiras do veículo que, para uma determinada configuração dos ângulos das articulações e suas respectivas variações (considerando certo horizonte) promova uma variação requerida para o ângulo entre os dois últimos *trailers*. A questão relativa à obtenção da inversa do modelo aproximado do sistema que se deseja controlar, com horizonte de predição característico, é também objeto de estudo e análise neste trabalho.

Em suma, a dissertação apresenta uma nova sistemática para o desenvolvimento de preditores e controladores neurais completos, com horizonte temporal fixo, utilizando redes neurais *feedforward* multicamadas estáticas, na descrição dos movimentos à ré de robôs móveis multiarticulados, no espaço de configuração. Aliado a isso, tendo em vista todas as etapas do projeto, uma ferramenta original, referida como uma *Graphical User Interface*[12], é desenvolvida com o intuito de auxiliar o processo de criação, treinamento e validação de diversas topologias de redes e de atribuição de seus parâmetros para o treinamento. Além disso, essa ferramenta vem facilitar a determinação de um horizonte de predição, para um robô, que gere melhores resultados nos processos de modelagem e controle, considerando o fato de essa ser uma tarefa exaustiva. A *Interface* foi implementada usando o *software* Matlab® a fim de incorporar as funcionalidades dos seus *toolboxes*. A sistemática proposta e as ferramentas desenvolvidas são gerais, aplicáveis não só ao protótipo e à arquitetura do RMMA em questão.

Existem na literatura algumas propostas de técnicas de modelagem e controle do movimento à ré de robôs móveis. No entanto, grande parte da tecnologia desenvolvida está voltada para carros de passeio ou para veículos com apenas um único *trailer*. De fato, apenas um número reduzido de trabalhos utiliza um protótipo de veículo multiarticulado real, como o utilizado nesse trabalho. A abordagem neural, em situações reais, é restrita a veículos não articulados (Demcenko, 2008), porém, em alguns casos, se utilizam veículos articulados com somente um *trailer*, mas validados usando dados obtidos somente por simulação (Kinjo, 2006), o que ilustra a originalidade da proposta dessa pesquisa.

No presente trabalho, tem-se uma aplicação real em um protótipo multiarticulado com forte restrição no ângulo de controle. Observa-se que essa restrição, à priori, pode simplificar a obtenção de preditores, mas pode complicar muito a síntese de controladores, pois aumenta as restrições de configurações críticas. De qualquer forma, o problema é original e os resultados são inéditos, pois a síntese de redes neurais estáticas na aproximação de preditores

[12] A interface foi desenvolvida fazendo uso do *Guide* do *software* Matlab®.

ou controladores completos, nas condições aqui estabelecidas, ainda não foi abordada na literatura.

Finalizando, esse trabalho se situa no universo de problemas de navegação ou manobras dos RMMA's caracterizados no espaço de configuração. Seu escopo se restringe apenas ao estudo, análise e proposição de uma metodologia empregável no nível de controle. Caberá aos trabalhos futuros uma discussão aprofundada sobre o nível de supervisão, sua integração ao nível de controle e a implementação prática de ambos no protótipo utilizado.

1.7 Organização desta Dissertação

A dissertação propõe uma nova sistemática e novas ferramentas para o projeto de preditores e controladores neurais completos a horizonte fixo, utilizando redes estáticas *feedforward*, na descrição dos movimentos à ré de RMMA's, no espaço de configurações.

De forma a melhor organizar a descrição da metodologia empregada na solução do problema de modelagem e de controle dos movimentos à ré de RMMA's, nas condições previamente estabelecidas, e de maneira a permitir uma clara apresentação das ferramentas e das soluções implementadas, este trabalho encontra-se organizado em sete capítulos e dois apêndices descritos a seguir.

a) Capítulo 1: **Introdução**

Apresenta o assunto a ser investigado ao longo deste trabalho, contextualizando-o e justificando-o. Contém informações gerais que dizem respeito ao tema o qual se deseja explorar e apresenta, sob a forma de um breve histórico, as contribuições existentes na literatura. Finaliza descrevendo, de maneira geral, a metodologia a ser aplicada na busca de soluções para o problema de modelagem e controle do RMMA e apresentando os objetivos gerais e específicos a serem alcançados.

b) Capítulo 2: **O Sistema de Representação do Robô Móvel Multiarticulado**

Discute a representação empregada no trabalho para retratar a cadeia cinemática do robô móvel. Apresenta o protótipo, a ser utilizado na aquisição de dados e no posterior projeto dos preditores e dos controladores, detalhando seus componentes estruturais e suas funcionalidades.

c) Capítulo 3: **Redes Neurais**

Relata a importância do estudo das Redes Neurais como solução na área do Controle Inteligente. Apresenta uma visão teórica detalhada das redes neurais, destacando seus conceitos, seus componentes, suas características e propriedades a fim de modelar o funcionamento do cérebro humano. Além disso, detalha os procedimentos envolvidos no projeto de uma rede neural, explicando as idéias de treinamento, aprendizado e validação. Por fim, apresenta, em geral, as vantagens e os fatores restritivos do uso das redes neurais e mostra exemplos de aplicações dessas estruturas em diversas áreas do conhecimento.

d) Capítulo 4: **Modelagem Analítica**

Esse capítulo apresenta uma modelagem para as condições singulares de giro e para os ângulos críticos, extraídos de um modelo analítico original das equações de movimento à ré de RMMA's, no espaço de configurações, desenvolvido por Ferreira (2011) e com sistemática de obtenção detalhada por Pandolfi (2011). Esse modelo é validado, em condições de operação real, para o protótipo de RMMA utilizado. Finaliza, apresentando as formulações matemáticas dos modelos e apresentando os resultados obtidos a partir dessas expressões.

e) Capítulo 5: **O Problema de Predição: Modelagem Neural do Robô Móvel Multiarticulado**

Discute a etapa imediatamente anterior ao projeto do controlador neural: a modelagem neural do robô, via preditores estáticos completos, com horizonte de predição fixo. Inicialmente, apresenta detalhes sobre a fase de obtenção de dados para o treinamento de uma rede neural, discutindo a estratégia empregada para a coleta dos mesmos, face às premissas e restrições do modelo, além dos mecanismos aplicados às informações adquiridas a fim de prepará-las devidamente de modo a formarem um conjunto de dados apropriado para o treinamento. Na sequência, para cada preditor, analisa a topologia utilizada, sendo feita as justificativas necessárias para a escolha da mesma, cita alguns procedimentos adotados, fundamentais a garantir um treinamento adequado e satisfatório, e analisa a parametrização escolhida quando da realização de tal processo. Por fim, apresenta as metodologias esquematizadas para o processo de validação de cada preditor neural projetado e analisa os resultados alcançados, a fim de determinar o horizonte de predição que caracterizará o preditor com melhor desempenho e de determinar os limites válidos, em simulação, do modelo escolhido a servirem de base à validação do correspondente controlador em malha

aberta. Ao longo desse capítulo, são mostradas algumas funcionalidades da *Interface* criada, relativas às etapas nele discutidas.

f) Capítulo 6: **O Problema de Controle**

Discute a estratégia empregada no projeto do controlador neural completo, com horizonte fixo. Apresenta duas formas de obtenção de controladores neurais inversos: indiretamente, a partir do preditor escolhido ou diretamente a partir de dados, porém os experimentos são feitos com base somente no último caso (o capítulo deixa a implementação de controladores indiretos a cargo de trabalhos futuros, fazendo as devidas justificativas). Apresenta o processo de treinamento do controlador neural direto e simula o processo de validação do mesmo em malha fechada, usando, para isso, dados também extraídos do sistema real. Durante sua validação, o desempenho do controlador neural sintetizado é comparado quando colocado em malha tanto com o preditor neural (oriundo da etapa de modelagem retratada no Capítulo 5) quanto com o preditor analítico (sintetizado a partir do modelo apresentado no Capítulo 4). Os resultados atingidos em cada etapa são mostrados e analisados. Ao longo desse capítulo, são mostradas as funcionalidades da *Interface* implementada correspondentes à etapa de controle. Finaliza apresentando abordagens alternativas a serem empregadas para lidar com o problema de controle do robô móvel multiarticulado e as estratégias adotadas para implementá-las, a fim de se obter resultados ainda melhores.

g) Capítulo 7: **Conclusões**

Retoma os objetivos específicos e gerais do trabalho, analisando os sucessos, as contribuições deixadas e os imprevistos (insucessos) observados, ressaltando, nesse último caso, as abordagens alternativas propostas durante o trabalho. Destaca os pontos satisfatórios do projeto e aqueles que não foram tratados da maneira mais adequada e que, portanto, merecem uma maior investigação em trabalhos futuros. Encerra vislumbrando novos horizontes possíveis de pesquisa para prosseguimento a partir deste trabalho.

h) Apêndice 1: **O Algoritmo de Aprendizado *Backpropagation***

Descreve o funcionamento geral do algoritmo de treinamento neural *Backpropagation*, destacando as etapas do mesmo a fim de se realizar o ajuste necessário dos parâmetros estruturais (pesos e *biases*) de uma rede de modo a aproximar a reposta real daquela desejada, para uma determinada entrada. Além disso, apresenta algoritmos que são provenientes de variações desse método.

i) Apêndice 2: **Condições Singulares de Giro**

Mostra o arquivo contendo as condições de giro, a partir do qual são extraídos os dados que irão ajudar a compor a base de dados de cada preditor e controlador neurais com horizonte de predição característico.

j) Anexos: **DVD**

Em anexo a esta dissertação, é disponibilizada um DVD contendo todo o código da *Interface* criada. Esse engloba as principais rotinas implementadas, usando o *software* Matlab®, para manipulação dos dados e para criação, parametrização, treinamento, teste, validação e simulação das redes neurais. Além disso, contém um vídeo explicativo do uso da *Interface*.

Capítulo 2: O Sistema de Representação do Robô Móvel Multiarticulado

Na busca de uma solução para o problema de controle, faz-se necessária a prévia compreensão do sistema antes que se possa associar tal conhecimento ao desenvolvimento de um modelo ou de um controlador para o mesmo.

O conhecimento da dinâmica do sistema a ser modelado assume um caráter imprescindível quando da fase de projeto e validação do modelo correspondente, bem como no início do projeto de qualquer controlador. Só através da clareza adequada do comportamento do sistema é que se torna possível detectar imperfeições ou inconsistências na modelagem, assim como deficiências na estratégia de controle implementada.

Independentemente das soluções já propostas na literatura e daquelas em andamento para o controle de robôs móveis, considerando a dificuldade em se trabalhar com as plantas a serem controladas em sua escala real, faz-se necessária a existência de um protótipo, de grande similaridade, guardadas as devidas proporções, a um veículo multiarticulado real, que auxilie na viabilização, construção, validação e aprimoramento de tais soluções. Além disso, mesmo sendo possível se obter boas aproximações e modelos em simulações computacionais destas plantas, muitos fatores e distúrbios são inerentes às plantas reais, se tornando muito difíceis ou até impossíveis de serem simulados.

Especificamente, para o presente trabalho, a partir do protótipo utilizado será possível extrair o conhecimento necessário à modelagem do RMMA e à validação do modelo correspondente e, posteriormente, à obtenção do controlador neural. Nesse capítulo, são abordados, portanto, a representação da cadeia cinemática e a caracterização do protótipo a ser empregado na síntese e comprovação experimental dos modelos neurais e analíticos e do controlador.

2.1 Representação da Cadeia Cinemática do Robô Móvel Multiarticulado

Sob o ponto de vista prático, um robô móvel multiarticulado (RMMA) é caracterizado por sua cadeia mecânica articulada, em geral, com o primeiro elemento motorizado (*truck-tractor* ou cavalo mecânico), cuja direção é controlada pelas rodas dianteiras e a tração, pelas rodas traseiras ou dianteiras, acoplado a elementos passivos articulados (*trailers* ou semi-reboques) sem motorização ou controle local e com ligação (*hitching) on-axle* ou *off-axle* dos seus engates.

Sob a óptica da Cinemática, desconsiderando a influência da massa e atuação das forças, um robô móvel multiarticulado é uma cadeia cinemática aberta[13] composta por partes físicas individuais chamadas ligações (*links*)[14], onde cada uma delas encontra-se unida a seu precedente e/ou subsequente por uma junta revoluta[15] plana.

Focando no arranjo da cadeia cinemática, é possível, através do conhecimento das variáveis de configuração definir a qualquer instante a configuração da composição. Todavia, há de se analisar mais detalhadamente a questão da localização dos engates antes de se propor uma configuração genérica para a cadeia.

De uma maneira geral, levando em consideração os diversos veículos multiarticulados que se pode encontrar e tomando como critério a posição dos engates e, portanto, a localização das articulações, poder-se-ia dividi-los em dois casos principais: com engate sobre o eixo traseiro (*on-axle hitching*) e fora do mesmo (*off-axle hitching*), como mostra a Figura 2.1.

(a) (b)

Figura 2.1 Tipos principais de posicionamentos de engates. (a) *Link* integrante da cadeia cinemática com engate sobre o eixo traseiro (acoplamento sobre o eixo). (b) *Link* integrante da cadeia cinemática com engate fora do eixo traseiro (acoplamento deslocado do eixo).

[13] Cadeia cinemática aberta: cadeia cinemática onde um *link* pode estar conectado a apenas uma junta.

[14] Ligações (*links*): correspondem às partes individuais físicas que formam coletivamente a composição do veículo ou o corpo do robô móvel multiarticulado.

[15] Juntas revolutas: são juntas que estabelecem como restrição, às ligações que compõem o robô, o movimento de rotação em torno de si.

Na Figura 2.1(a), é possível observar uma configuração de engate muito comum em veículos multiarticulados de carga, conhecidos como *Bitrens* ou *Bi-tanks*, como os da Figura 2.2. Nesse tipo de veículo, o engate se localiza sobre o eixo traseiro do respectivo *trailer* da frente. A Figura 2.1(b), por sua vez, representa o acoplamento encontrado em veículos multiarticulados onde os engates estão situados em pontos deslocados do eixo traseiro (exemplo real na Figura 2.3).

(a)

(b)

Figura 2.2 Aplicação de *trailers* com engate sobre o eixo. (a) *Bitrem*. (b) *Bi-tank*.

Figura 2.3 Aplicação de *trailers* com engate fora do eixo.

Há ainda a possibilidade de um terceiro arranjo, no qual teríamos uma composição híbrida, possuindo simultaneamente ligações com engates sobre os eixos traseiros e fora deles. Como exemplos práticos deste tipo de configuração, têm-se a composição utilizada no transporte de eucaliptos e aquela que constitui o protótipo utilizado neste trabalho, onde o engate do primeiro *trailer* ao *truck* localiza-se sobre o eixo traseiro do mesmo, enquanto o engate entre os *trailers* é acoplado a pontos fora dos eixos traseiros desses.

Assim, considerando-se o anteriormente proposto, a cadeia cinemática genérica, utilizada no estudo do problema de modelagem e controle de um RMMA em movimentos à ré, é esquematizada na Figura 2.4. Na representação de um RMMA, o vetor de configuração tem como componentes os sucessivos ângulos, θ_i, entre os respectivos elementos da cadeia. A direção das rodas dianteiras é caracterizada pelo ângulo γ. Os parâmetros geométricos longitudinais do i-ésimo elemento da cadeia do RMMA, A_i e B_i, ilustrados na Figura 2.4, são definidos por: A_1 como sendo a distância entre os eixos do *truck*; A_i (i > 1), distância do ponto de engate dianteiro ao eixo; e B_i, distância do ponto de engate traseiro ao eixo (para $B_i \neq 0$, tem-se *off-axle hitching*).

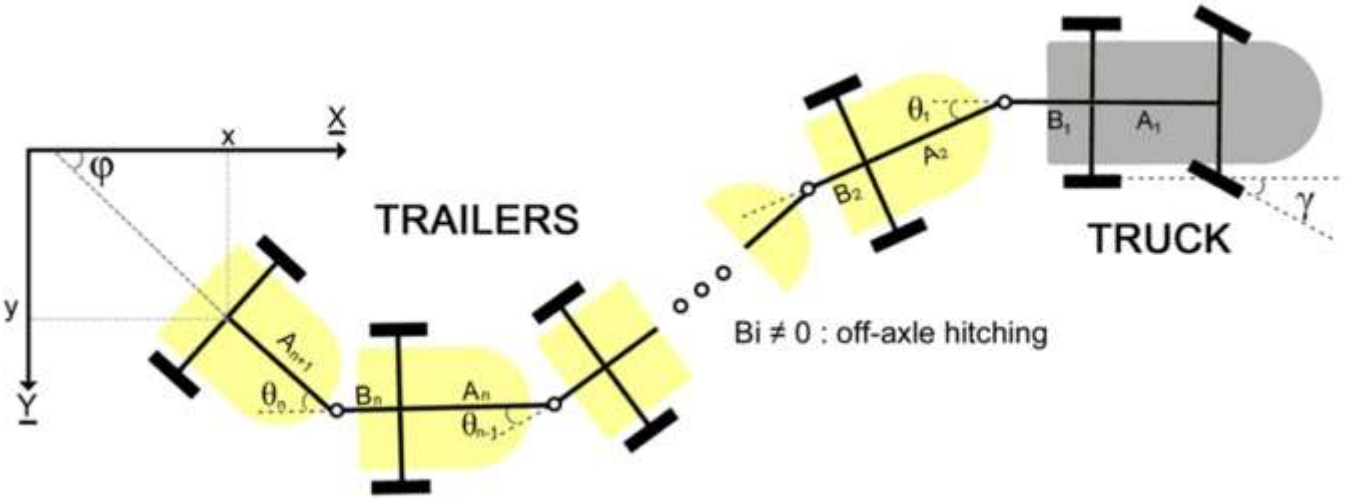


Figura 2.4 Representação da cadeia cinemática genérica de um RMMA.

Esta representação apresenta uma cadeia de *L* graus de liberdade ($L = n + 1$, mantida a velocidade constante – onde *n* é a dimensão do vetor configuração) sendo, por isso, designada como uma cadeia complexa.

Um veículo multiarticulado de três graus de liberdade pode ser mais comumente representado por um caminhão com uma composição formada por um *truck* e dois *trailers*, conforme a Figura 2.5. Essa configuração é de grande utilização, pois consegue manter uma boa relação de compromisso entre o volume de carga transportado e a dificuldade de

manobra. Além disso, essa é suficiente para manter a generalidade do problema de manobras. Um maior número de elementos passivos (*trailers*) pode dificultar e inviabilizar manobras ou testes (no caso do uso de protótipos) em ambientes espacialmente restritos.

Figura 2.5 Veículo multiarticulado de três graus de liberdade.

Logo, tendo em vista os fatos anteriores, a cadeia cinemática de interesse, do RMMA utilizado neste trabalho, é restrita a três graus de liberdade. Além disso, é constituída de tração traseira e engate *trailer-trailer* com ligação *off-axle* e engate *truck-trailer*, *on-axle*. A Figura 2.6 mostra a cadeia cinemática do protótipo do RMMA em questão e os valores de seus parâmetros geométricos (A_i, B_i), evidenciando o ângulo da direção (γ) e o vetor de configuração (θ_1, θ_2).

Uma vez mostrada a descrição da cadeia cinemática, será apresentado, na seção seguinte, o protótipo utilizado na geração do preditor e controlador neurais, bem como, na validação dos mesmos. Maiores detalhes a respeito do desenvolvimento dessa plataforma experimental são encontrados em Barreto (2010). Esse possibilitou o desenvolvimento e o aprimoramento de uma plataforma que permitisse a implementação, a validação e a análise de desempenho de soluções de controle obtidas por meio de diferentes métodos e estratégias.

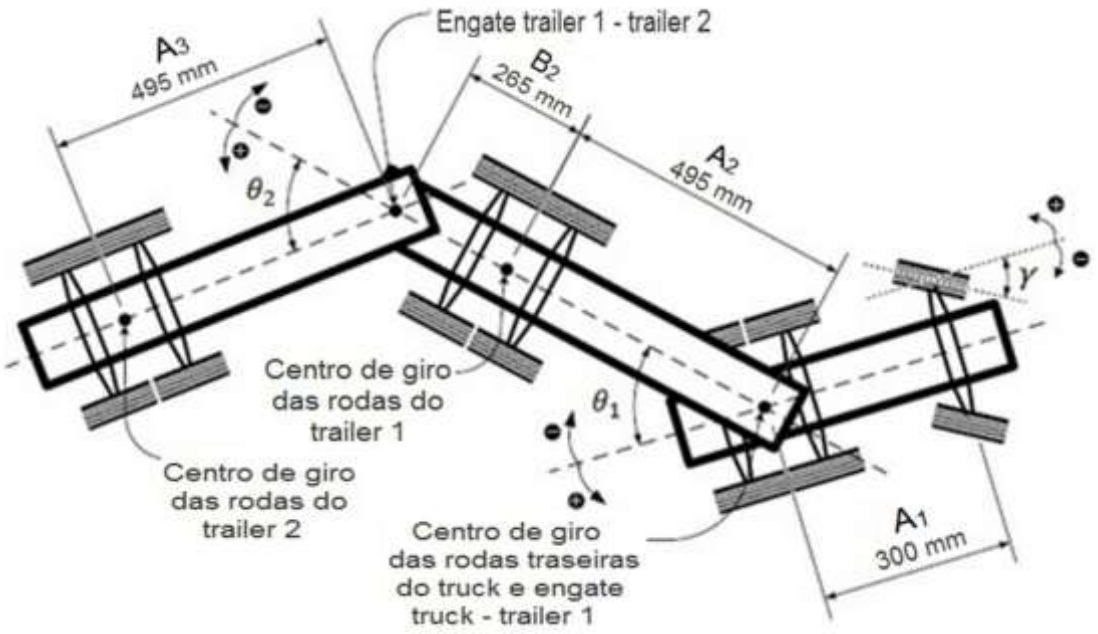


Figura 2.6 Representação da cadeia cinemática do RMMA utilizado.

2.2 Descrição do Protótipo Utilizado

Para os experimentos, foi utilizado um protótipo de um veículo multiarticulado com algumas características mecânicas e elétricas particulares que valem destacar, tais como:

a) Apresentar derrapagem desprezível ou nenhuma ao efetuar curvas;
b) Possibilitar a obtenção dos valores dos ângulos das articulações;
c) Ser passível de ser desmontado e transportado de maneira rápida e fácil;
d) Possibilitar, de maneira simples, mudanças na configuração da cadeia cinemática, no que se refere à ampliação ou redução do grau de liberdade.

Conforme comentado, este protótipo destinou-se, basicamente, a auxiliar na análise do problema de controle e no fornecimento de dados práticos (além das condições singulares de giro) necessários ao seu modelamento via redes neurais e ao posterior projeto do controlador.

2.2.1 O *Truck* e os *Trailers*

Neste trabalho foi utilizado um protótipo de RMMA de três graus de liberdade, constituído por um elemento trator (veículo de tração) e dois *trailers*, como mostrado na

Figura 2.7. O automodelo mede no total 1.73*m*, quando alinhado, e foi produzido pela empresa Tamiya® Co. Ltd, baseado na réplica do caminhão *Globe Liner*, em escala 1/14.

Figura 2.7 Protótipo multiarticulado utilizado.

Além disso, o elemento trator é dotado de um sistema de transferência de tração do tipo diferencial e de suspensão independente com feixe de molas e amortecedores, o que facilita o transporte extra de peso sem prejudicar sua movimentação, sobretudo em pisos irregulares. A tração da composição é provida por um motor DC. Acoplado ao motor, existe uma caixa de redução, ou câmbio, que possibilita a escolha de três tipos de relação de engrenagens, representando a primeira, a segunda e a terceira marchas.

Os semi-reboques utilizados consistem em dois *Flatbed semi-trailers* também de escala 1/14 da marca Tamiya®. Esses também apresentam amortecimentos independentes para as rodas, o que assegura maior estabilidade nas curvas, e possuem rolamentos nos eixos de tração contribuindo para um comportamento dinâmico mais realístico.

2.2.2 Os Engates e os Potenciômetros

Um ponto importante reside na implementação dos engates e no acoplamento de potenciômetros a estes. No protótipo utilizado, há um misto das configurações básicas de acoplamento possíveis, conforme mencionado anteriormente. A ligação do engate entre o *truck* e o primeiro *trailer* é do tipo *on-axle hitching*, conforme a Figura 2.8(a); já a do engate entre o primeiro e o segundo *trailers* é *off-axle hitching*, conforme a Figura 2.8(b).

Os engates foram implementados pela Automática Tecnologia® de modo a adequar-se aos requisitos de projeto, fazendo uso de potenciômetros de instrumentação acoplados às articulações. Essa mesma empresa forneceu e instalou os potenciômetros no robô.

(a) (b)

Figura 2.8 Localização dos potenciômetros e dos engates do protótipo. (a) Engate entre o *truck* e o primeiro *trailer*: *on-axle hitching*. (b) Engate entre os *trailers*: *off-axle hitching*.

A leitura da configuração da composição, composta pelos ângulos em suas articulações, é feita, ao longo das manobras, por meio de sensores embarcados, tais como potenciômetros de alta resolução instalados em cada engate do robô. Os valores nominais desses potênciometros são de $20K\Omega$ e a precisão é de 1%. Conforme será discutido na próxima seção, os valores lidos pelos potenciômetros são inicialmente salvos localmente e então é feito o *download* para um computador remoto através do sistema de comunicação.

2.2.3 O Sistema Embarcado

O sistema embarcado do protótipo contém um microcontrolador *Atmega32* da *Atmel Products* com 32*KB* de memória *flash*, 2*KB* de memória RAM, 40 pinos e até 16*MHz* de frequência de processamento. Por sua velocidade e memória maiores que as de outros microcontroladores similares, baixo custo, baixo consumo de energia, programador gratuito e facilidade de obtenção, o *Atmega32* é uma ferramenta adequada para aplicações de robótica móvel.

O sistema embarcado está basicamente relacionado ao acionamento do mecanismo de direção, sensoriamento de modo geral, torque, controle de velocidade, seleção de marcha do câmbio, conversão de sinais analógicos em digitais, comunicação, temporização de eventos e armazenamento de dados. O controle de alto nível é realizado de modo *off-board*, ou seja, todo o processamento mais complexo destinado ao controle de manobras e determinação de trajetórias pode ser convenientemente executado por outro computador de maior desempenho

e melhores recursos computacionais, que ao final, transmite a ação de controle ao circuito de acionamento do veículo. A Figura 2.9 exibe o sistema embarcado juntamente com seus principais componentes.

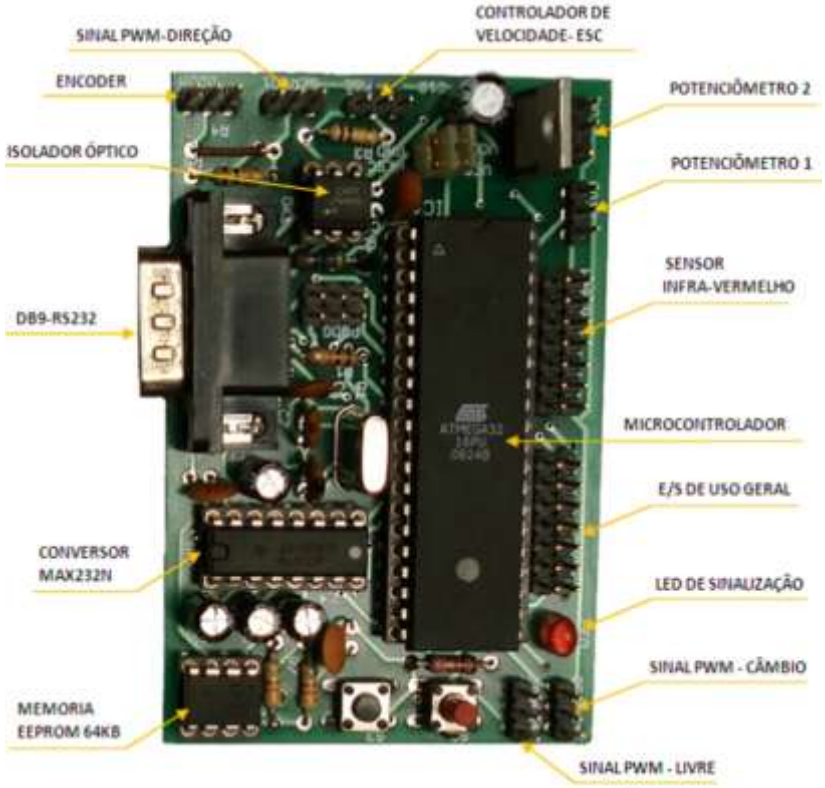


Figura 2.9 Principais componentes do sistema embarcado.

Uma das finalidades do microcontrolador consiste em receber do computador os comandos de velocidade linear e de direção e aplicar sinais de PWM ao servo motor e ao *E.S.C. MC230CR* (*Eletronic Speed Control* modelo *MC230CR*), fabricados pela Futaba® e responsáveis pela direção e velocidade, respectivamente. O acionamento do motor do robô é realizado pelo *E.S.C.*, que é um circuito eletrônico CMOS responsável por receber o sinal de referência, do tipo PWM, e convertê-lo em outro sinal de potência, também PWM, porém com tensão e frequência apropriadas para serem aplicadas ao motor. Através da variação do ciclo de trabalho do sinal PWM de referência, o *E.S.C.*, por meio de uma ponte H existente em seu circuito, controla o sentido de giro e a potência fornecida ao motor e, consequentemente, varia sua velocidade. O controle mais refinado da velocidade também conta com o auxílio de um encoder conectado a uma das portas A/D do *Atmega32*.

Além disso, o microcontrolador é capaz de ler os valores dos potenciômetros acoplados aos engates do protótipo e de armazená-los localmente ou enviá-los ao computador

remoto. Com isso, mantém interface com um módulo de rádio frequência para envio e recepção das variáveis a serem controladas.

Objetivando-se expandir a capacidade de armazenamento de dados além da capacidade do microcontrolador, foi instalado um C.I. (Circuito Integrado) de memória EEPROM, capaz de armazenar até 64*KB*, e, posteriormente, de memória *flash*. A armazenagem *on-board* (local) de dados é mais rápida e mais segura, pois evita a necessidade da transferência imediata dos dados coletados para o computador remoto durante o experimento. Assim, ao término dos experimentos, os dados ficam ainda armazenados na memória e, então, podem ser transferidos quando conveniente.

Por fim, todo o circuito eletrônico é alimentado por uma bateria recarregável com tensão nominal de 7.2*volts* e capacidade de 2500*mAh*, que garante uma autonomia média de 25 minutos de operação ao robô, sem desligá-lo. Essa mesma bateria é utilizada no acionamento do motor.

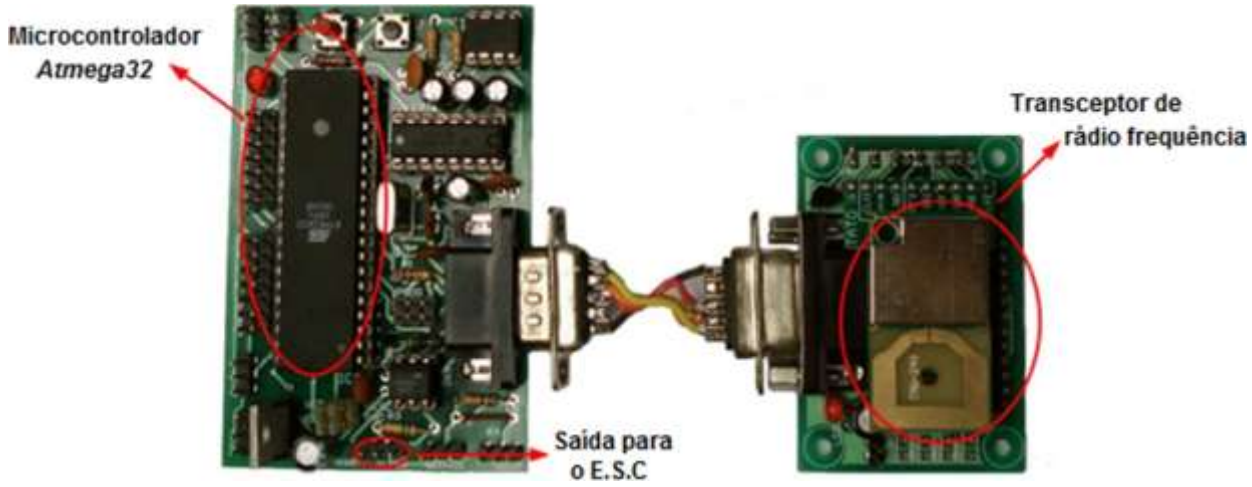


Figura 2.10 Sistema embarcado e transceptor de rádio frequência.

2.2.4 O Sistema de Comunicação

O sistema de comunicação foi implementado com transceptores, modelo *TRW-2.4* fabricado pela *Laipac Tech Inc*., operando na frequência de 2.45*GHz* e permitindo transferência de dados a taxas entre 250*kbps* e 1*Mbps*.

Tanto o computador quanto o microcontrolador utilizam um sistema de comunicação em rádio frequência (RF) para transmitir e receber dados entre si. Assim, foram utilizadas duas placas semelhantes: a primeira, mostrada na Figura 2.10, para ser conectada ao *hardware* embarcado e a segunda, mostrada na Figura 2.11, conectada ao computador.

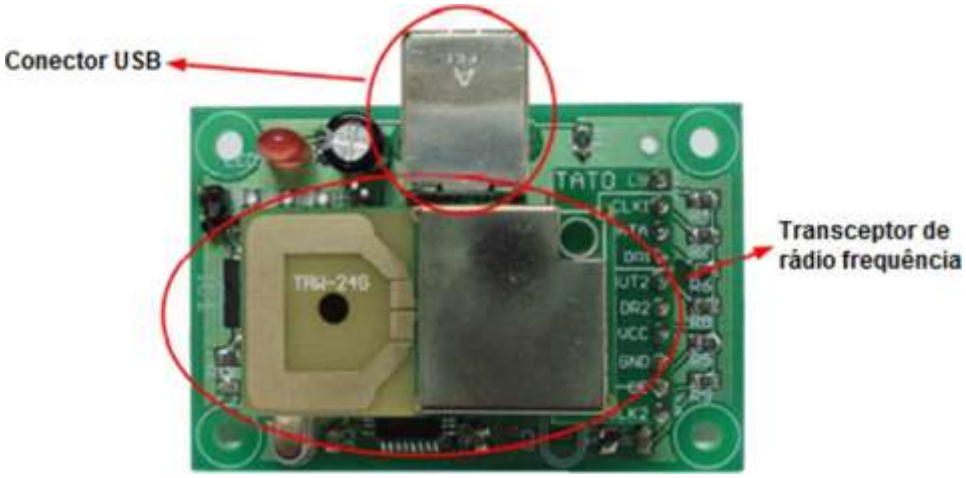


Figura 2.11 Sistema de comunicação RF com entrada USB para conexão com computador.

2.2.5 O Sistema de Odometria

O sistema de odometria é composto de um *encoder* óptico do tipo incremental que possibilita a determinação da distância percorrida e da posição (através da relação entre pulsos elétricos e voltas no eixo de rotação) e da velocidade do robô móvel (quando se considera a quantidade de pulsos gerados em certo intervalo de tempo).

O *encoder* se baseia no princípio da reflexão da luz. Um fototransistor é excitado com a presença de luz; assim, utilizando-se um disco com partes claras e escuras, é possível gerar um sinal de luz pulsado que altera o modo de operação do fototransistor entre as regiões de corte e saturação, gerando um sinal elétrico pulsado correspondente.

O número de pulsos por volta é contado por uma variável de 8 *bits* do microcontrolador. Isso faz com que o intervalo de contagem seja de 0 a 255. Ao identificar corretamente o estouro do contador, que ocorre quando o valor de contagem passa de 255 para 0, o computador remoto adiciona 256 unidades a cada valor corrente, após o estouro, para se obter a contagem total dos pulsos. A detecção do estouro se faz de maneira trivial, observando-se o instante em que o valor da leitura atual dos pulsos é menor do que o valor lido anteriormente.

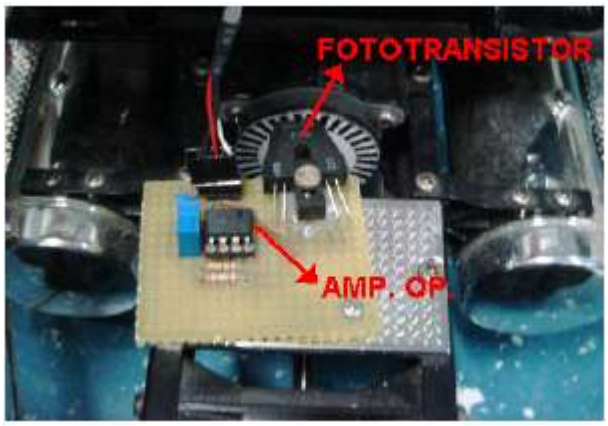


Figura 2.12 Local de instalação do *encoder*, o qual compõe o sistema de odometria.

Finalmente, no capítulo seguinte será discutido todo o embasamento teórico sobre as Redes Neurais, fundamental para a posterior etapa de modelagem do RMMA e projeto do controlador.

Capítulo 3: Redes Neurais

As Redes Neurais Artificiais constituem a base do Controle Inteligente de Sistemas. Tais estruturas modelam aproximadamente parte da arquitetura do cérebro humano e possuem propriedades de aprender, assimilar, lembrar, errar e reaprender com seus erros. Na divisão de pesquisa e desenvolvimento, as Redes Neurais fazem parte da Inteligência Artificial na sua vertente cognitiva. Na Figura 3.1, é definida a sua situação no escopo da Inteligência de Máquinas e da Inteligência Computacional.

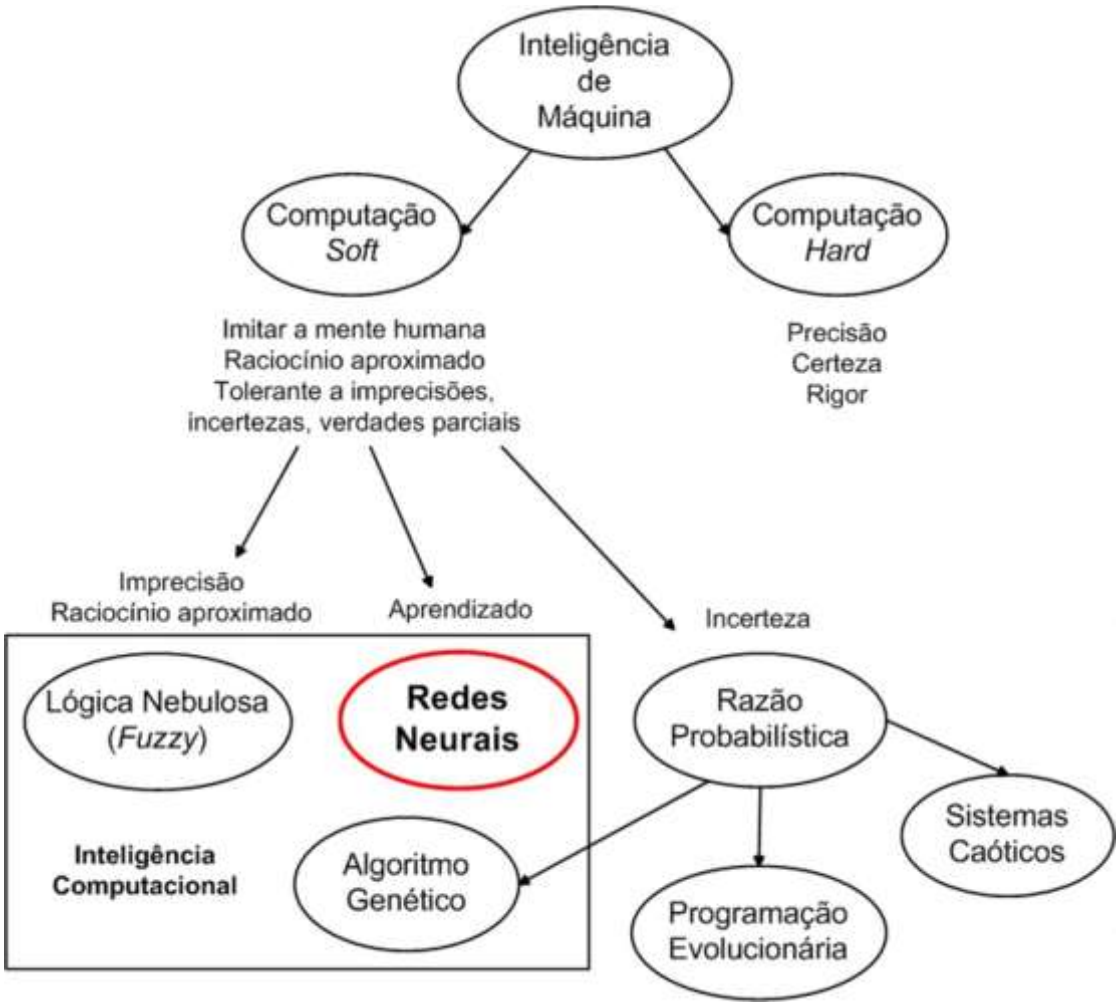


Figura 3.1 Delimitação do escopo das Redes Neurais na Inteligência Artificial.

Pesquisas no campo de Redes Neurais têm atraído crescente atenção nos últimos anos. Desde 1943, quando Warren McCulloch e Walter Pitts apresentaram o primeiro modelo de neurônios artificiais, novas e mais sofisticadas propostas têm sido feitas (Rojas, 1996).

Os trabalhos com redes neurais têm sido motivados, à princípio, pela presença nestas das capacidades de aquisição e manutenção do conhecimento e de reconhecimento, próprias do cérebro humano e que em muito diferem da maneira pela qual os computadores digitais convencionais processam as informações. O cérebro humano é um computador paralelo, não-linear e altamente complexo (Haykin, 1999). A capacidade do cérebro em resolver problemas, com os quais os computadores ainda não podem eficientemente lidar, é maior (Rojas, 1996).

O poderio de processamento das redes neurais reside principalmente em sua estrutura extremamente paralela, hierárquica, redundante, massivamente distribuída e em sua habilidade de aprender e, por conseguinte, generalizar. As redes neurais são sistemas, dos quais as estruturas são parcialmente pré-determinadas. O ajuste de alguns parâmetros modifica a capacidade da rede em achar a melhor combinação para a solução de um determinado problema. Assim, as redes neurais representam sistemas adaptativos, auto-organizáveis, assim como cada neurônio constituinte.

Biologicamente falando, o mecanismo de produção e transporte de sinais de um neurônio a outro é um fenômeno fisiológico bem compreendido, mas como esses sistemas individuais cooperam para formarem estruturas paralelas e complexas com grande capacidade de processamento, armazenamento e transmissão de informação ainda não foi completamente elucidado. As Redes Neurais Artificiais integram uma geração promissora de sistemas de processamento de informações.

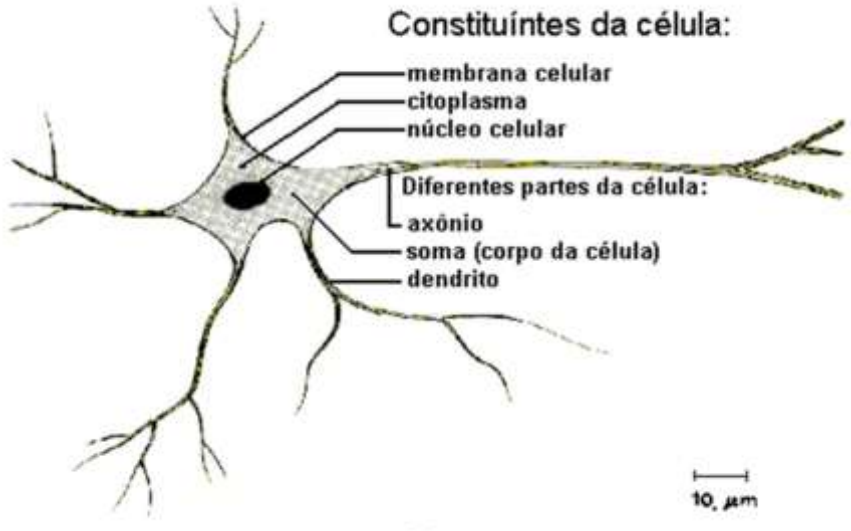
Outra característica bastante atrativa e fortemente explorada das redes artificiais é sua capacidade de aproximar uma enorme gama de funções não-lineares. É notadamente crescente o emprego dessas estruturas no modelamento de sistemas dinâmicos complexos não-lineares, melhorando o desempenho ou a inteligência dos mesmos até em ambientes com imprecisão, incerteza ou ruído. Além disso, merece destaque o emprego dessas estruturas na predição do comportamento de um determinado sistema, como na aplicação abordada no presente trabalho. Nessas circunstâncias, o objetivo consiste em estimar, após um horizonte definido, valores futuros de um processo levando-se em consideração diversas medidas prévias observadas em seu domínio.

Este capítulo destina-se a apresentar alguns dos fundamentos teóricos ligados às redes neurais artificiais, que permitirão uma melhor compreensão da aplicação das mesmas na obtenção do modelo do robô multiarticulado ao efetuar movimentos à ré, bem como, no projeto do controlador.

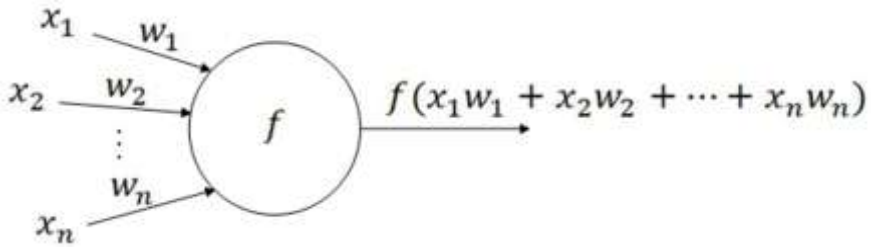
3.1 Neurônios Biológicos e Neurônios Artificiais

O sistema nervoso é formado por um conjunto extremamente complexo de células amplamente interconectadas, os neurônios. Eles têm um papel essencial na determinação do funcionamento e comportamento do corpo humano e do raciocínio. Um típico neurônio biológico, vide Figura 3.2(a), é formado por três principais partes: o corpo central (ou soma), onde o núcleo celular está situado; os dendritos, que podem ser entendidos como um conjunto de terminais, conectados ao corpo celular, de entrada de estímulos nervosos vindos de diversos outros neurônios ou do próprio meio externo; e os axônios que são longos terminais de saída.

A comunicação entre essas células nervosas se dá através de sinapses. Sinapse é a designação da região onde dois neurônios entram em contato e através da qual os impulsos nervosos são transmitidos entre eles. Deste modo, os impulsos elétricos recebidos por um neurônio A, através de seus dendritos, em um determinado momento, são processados e, atingindo um dado limiar de ação (*threshold*), o neurônio A dispara, produzindo uma substância neurotransmissora que flui do seu corpo celular para o axônio, que pode estar conectado a um dendrito de outro neurônio B. O neurotransmissor, por sua vez, pode diminuir ou aumentar a polaridade da membrana pós-sináptica, inibindo ou excitando a geração dos pulsos no neurônio B (ou seja, o neurotransmissor pode aumentar ou diminuir o potencial elétrico dentro do corpo celular do neurônio receptor) – de acordo com tal característica, as sinapses podem ser classificadas como excitatórias ou inibitórias. Este processo depende de vários fatores, como a geometria da sinapse e o tipo de neurotransmissor.



(a)



(b)

Figura 3.2 Estrutura de um neurônio. (a) Neurônio Biológico (b) Neurônio Artificial.

Estima-se que uma rede neural biológica seja constituída por cerca de 100 bilhões de neurônios, cada um destes interligados, em média, a outros 6000 neurônios, o que gera um total de 600 trilhões de sinapses (Shepherd, 1990).

As redes neurais artificiais em muito se assemelham com suas contrapartes biológicas. Neurônios artificiais, também chamados de Unidades Computacionais (ou de Nós ou ainda simplesmente de Unidades), constituindo os elementos de processamento, possuem canais de entrada, um corpo central e um canal de saída. As sinapses (conexões ou *links*) são simuladas pelos pontos de contato entre o corpo celular e as conexões de entrada ou saída. Além disso, pesos são associados a esses pontos, os quais são responsáveis por armazenar o conhecimento.

A Figura 3.2(b) representa um neurônio artificial com *n* entradas[16]. Geralmente, cada canal de entrada tem um peso associado, w_i, o qual é multiplicado pelo dado recebido, x_i. Assim, a resposta de um neurônio é obtida com base no conjunto das informações integradas ao mesmo, sendo essas argumentos de uma função primitiva, f, escolhida arbitrariamente. Porém, essa função computada é, em geral, uma função de *n* parâmetros, sendo necessário que essa quantidade de argumentos seja reduzida em um único valor numérico. Assim, a função primitiva, conforme Figura 3.3, pode ser dividida em duas partes: uma função de integração, g, que combina a informação de fontes externas, transformando os *n* parâmetros em um único valor; e uma função de ativação, f, que produz a saída do neurônio, tomando esse único valor gerado como argumento. Geralmente a função de integração é representada por uma soma aritmética. A função de ativação limita a saída do neurônio dentro de um intervalo de valores razoáveis a serem assumidos por sua própria imagem funcional.

[16] *Fan-in*: conjunto de todas as combinações de entrada.

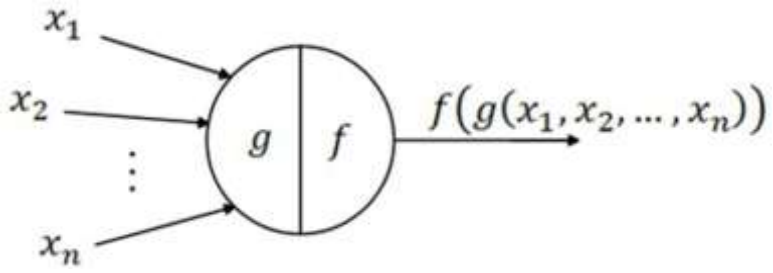


Figura 3.3 Funções de integração e de ativação de uma unidade computacional.

A Figura 3.4 mostra um neurônio, proposto por McCulloch e Pitts, inspirado nas características biológicas comentadas anteriormente, que é mais comumente chamado de Unidade M-P. Nesse modelo, a unidade computa uma soma ponderada de suas entradas $(x_1, x_2, \dots, x_m)$, tendo por base seus respectivos pesos $(w_1, w_2, \dots, w_m)$, e pode gerar dois tipos de saída:

a) A unidade gera como saída um sinal unitário (y = 1), se a soma for maior ou igual que um determinado *threshold* (θ). Nesse caso, diz-se que a unidade é excitada e dispara o sinal;

b) A unidade gera como saída um sinal nulo (y = 0), se a soma for menor que um determinado threshold (θ). Nesse caso, diz-se que a unidade é inibida e não dispara nenhum sinal.

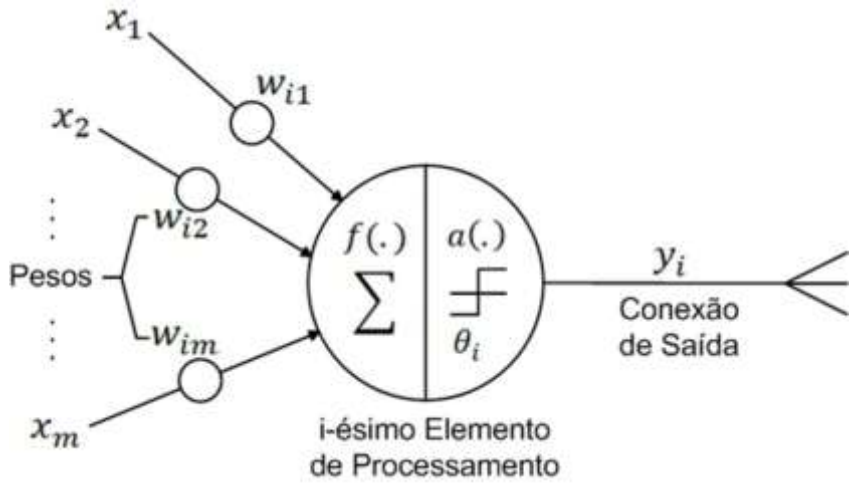


Figura 3.4 Processamento de uma unidade M-P.

Cada peso representa a força da sinapse, ou seja, de ligação de um neurônio a outro. Um peso positivo pode levar a uma excitação do neurônio, ou ainda, levar a sua inibição (nesse caso, a inibição é denominada *Absoluta*), conforme os critérios mencionados; já um peso negativo sempre leva a uma inibição da célula (a inibição passa a ser denominada *Relativa*). Um peso nulo significa a inexistência de conexão.

No caso em que o *threshold* é convertido em um peso correspondente a uma entrada adicional constante unitária da unidade, um limiar nulo deve ser levado em consideração. Esse peso passa a ser chamado *Bias*. Matematicamente, as entradas $\underline{x} = x_1, x_2, \dots, x_n$ correspondem ao vetor de entrada. O vetor de pesos é formado por números reais $\underline{w} = w_1, w_2, \dots, w_n$. Logo, pode-se denominar $\underline{\widehat{w}} = w_1, w_2, \dots, w_n, w_{n+1}$ como sendo o vetor de pesos estendido, onde $w_{n+1} = -\theta$, e $\underline{\hat{x}} = x_1, x_2, \dots, x_n, 1$, o vetor de entradas estendido. A condição de disparo de um neurônio é $\underline{\hat{x}} \cdot \underline{\widehat{w}} \geq 0$, quando $\underline{\hat{x}}$ e $\underline{\widehat{w}}$ correspondem aos vetores nas suas formas estendidas; ou $\underline{x} \cdot \underline{w} \geq \theta$, caso contrário. Uma unidade com tal caracterísitca pode ser visualizada na Figura 3.5.

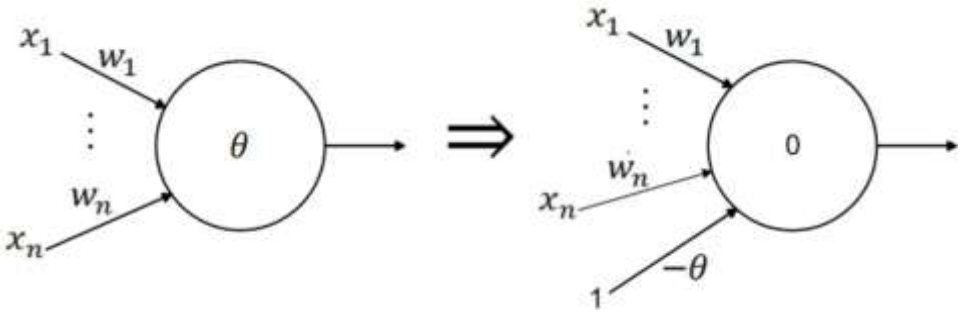


Figura 3.5 Unidade sem (à esquerda) e com (à direita) *Bias*.

Levando em consideração os fatos mencionados, pode-se repesentar a resposta da i-ésima unidade M-P, vide Figura 3.4, como sendo:

$$y_i = a\left(\sum_{j=1}^{m} \widehat{w}_{ij} \cdot \hat{x}_j\right) = a\left(\sum_{j=1}^{m} w_{ij} \cdot x_j - \theta\right) \tag{3.1}$$

onde a função de ativação $a(f)$ corresponde a uma função degrau:

$$a(f) = \begin{cases} 1, & se\ f \geq 0 \\ 0, & caso\ contrário \end{cases} \tag{3.2}$$

3.1.1 Funções de Integração e de Ativação de um Neurônio

Geralmente as funções de integração são representadas por uma função linear, conforme exemplificada na Equação 3.1. No entanto, funções mais complexas também podem ser consideradas, como mostradas a seguir.

a) Função Quadrática:

$$f = \sum_{j=1}^{m} w_j \cdot {x_j}^2 - \theta \tag{3.3}$$

b) Função Esférica:

$$f = \rho^{-2} \sum_{j=1}^{m} \left(x_j - w_j\right)^2 - \theta \tag{3.4}$$

onde ρ e w_j correspondem ao raio e ao centro da esfera, respectivamente.

c) Função Polinomial:

$$f_i = \sum_{j=1}^{m} \sum_{k=1}^{m} w_{ij_k} x_j x_k + x_j^{\alpha_j} + x_k^{\alpha_k} - \theta_i \tag{3.5}$$

onde α_j e α_k são constantes reais.

Já as funções de ativação mais comumente usadas, além da função degrau, mostrada anteriormente, são:

a) Função Degrau Bipolar (*Symmetric Hard Limiter*):

$$g(f) = sign(f) = \begin{cases} 1, & se \quad f \geq 0 \\ -1, & se \quad f < 0 \end{cases} \tag{3.6}$$

onde sign$(\bullet)$ corresponde à função de sinal.

b) Função Rampa (ou Linear):

$$g(f) = \begin{cases} 1, & se \ f > 1 \\ f, & se \ 0 \leq f \leq 1 \\ 0, & se \ f < 0 \end{cases} \tag{3.7}$$

c) Função Sigmoidal Unipolar[17] (ou Função Logística):

$$s_\lambda(f) = \frac{1}{1 + e^{-\lambda f}} \quad , \lambda > 0 \tag{3.8}$$

A constante λ pode ser selecionada arbitrariamente e, de acordo com seu valor, a forma da curva sigmóide pode ser mudada. Altos valores de λ aproximam a sigmóide à função degrau em torno da origem.

d) Função Sigmoidal Bipolar (também chamada Função Sigmoidal Simétrica ou Função Tangente Hiperbólica):

$$S(f) = 2\, s_\lambda(f) - 1 = \frac{2}{1 + e^{-\lambda f}} - 1 = \frac{1 - e^{-\lambda f}}{1 + e^{-\lambda f}}, \lambda > 0 \tag{3.9}$$

---

[17] Essa função possui como derivada: $s'_\lambda(f) = \frac{ds_\lambda(f)}{df} = \frac{\lambda e^{-\lambda f}}{\left(1+e^{-\lambda f}\right)^2} = \lambda\, s(f)\left(1 - s(f)\right)$. Ambas são abordadas no Apêndice 1, referente ao algoritmo de aprendizado neural, o *Backpropagation.*

As curvas de cada função apresentada são mostradas na Figura 3.6.

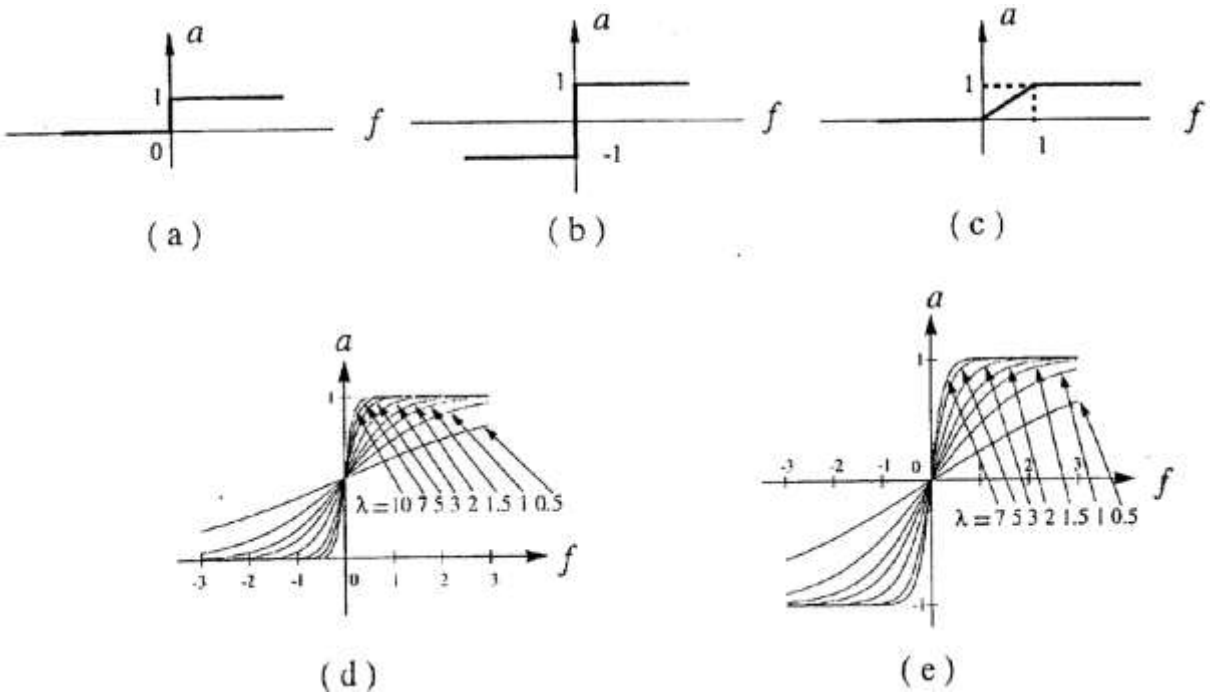


Figura 3.6 Funções de ativação. (a) Função Degrau (*Step Function*). (b) *Symmetric Hard Limiter*. (c) Rampa (*Ramp Function*). (d) Função Sigmoidal Unipolar. (e) Função Sigmoidal Bipolar. (Lim, 1996).

Neurônios artificiais com função de integração linear e função de ativação *hard limiter* são chamados *Linear Threshold Unit* (LTU); já os compostos por função de integração linear e função de ativação sigmoidal são chamados *Linear Graded Unit* (LGU). Ambos são os mais usados em redes neurais artificiais.

3.2 Definição Formal de Redes Neurais

Na seção anterior, analisamos as propriedades individualmente das unidades que irão compor uma rede neural. O próximo passo consiste em combinar esses elementos, observando o poder computacional da estrutura resultante, uma rede neural.

3.2.1 Arquitetura de Rede

Os menores elementos de quaisquer arquiteturas de rede são as unidades computacionais e suas interconexões. Cada unidade coleta informações oriundas de *n* entradas

com uma função de integração $\psi: \Re^n \rightarrow \Re$. A excitação total computada é, então, avaliada usando uma função de ativação $\phi: \Re \rightarrow \Re$.

Uma arquitetura de rede, segundo Rojas (1996), é representada por uma tupla (I, N, O, E) consistindo em um conjunto I de entradas, um conjunto N de unidades computacionais, um conjunto O de saídas e um conjunto E de conexões com pesos. Uma conexão, ou ramo, é uma tupla (u, v, w), onde $u \in$ I $\cup$ N, $v \in$ N $\cup$ O, e $w \in \Re$.

Arquiteturas de redes em camadas são aquelas nas quais o conjunto N de unidades computacionais é subdividido em l subconjuntos $N_1, N_2, \ldots, N_l$, formando as camadas da rede neural.

A região de entrada é aquela ligada ao subconjunto N_1 e definida apenas como ponto de onde as informações são provenientes, não consistindo em um conjunto de nós da rede e, portanto, não sendo realizado nenhum processamento nesse local. Tal região constitui a camada de entrada. A região da estrutura, formada pelo subconjunto N_l de nós de saída, representa a camada de saída da rede. As demais camadas da rede, correspondentes aos subconjuntos $N_2, \ldots, N_{l-1}$, são chamadas camadas intermediárias (também chamadas de ocultas ou escondidas). A rede mostrada na Figura 3.7 representa a arquitetura de rede discutida.

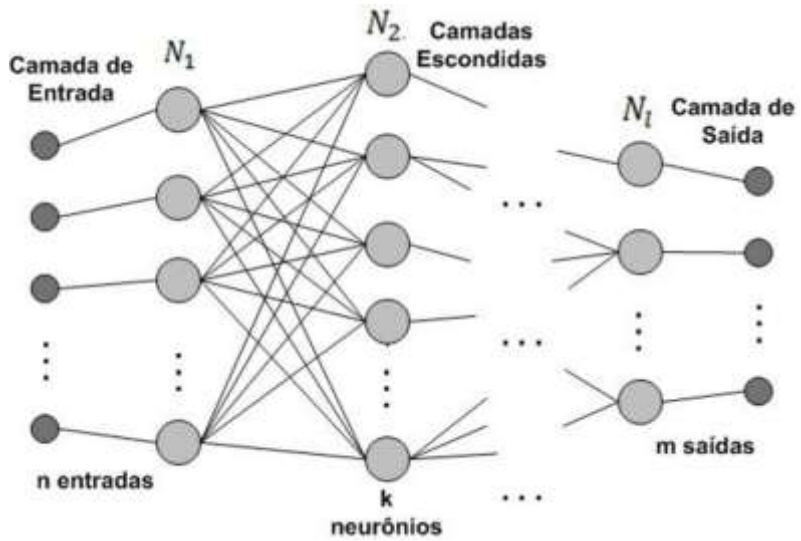


Figura 3.7 Uma arquitetura genérica de rede em camada.

3.2.2 Topologias de Redes Neurais

As redes neurais, em geral, independentemente da topologia empregada, se comportam como máquinas de mapeamento de entradas reais *n-dimensionais* $(x_1, x_2, \ldots, x_n)$ em saídas reias *m-dimensionais* $(y_1, y_2, \ldots, y_m)$, modelando, portanto, funções $F: \Re^n \rightarrow \Re^m$.

De acordo com a geometria das conexões de uma arquitetura de rede, podemos classificá-la em alguns tipos básicos de topologias, conforme será discutido a seguir.

Redes, nas quais a saída de um nó não constitui uma entrada de um outro nó da mesma camada ou de uma camada anterior, não formando ciclos, são chamadas redes *feedforward*. As entradas do sistema podem ser ligadas a vários nós, de modo a cada um gerar uma saída correspondente, conforme Figura 3.8(a). Dizemos que essa rede representa uma topologia do tipo *single-layer feedforward*. Quando várias camadas são interconectadas, essas formam uma rede do tipo *multi-layer feedforward*, como na Figura 3.8(b). A rede é dita estar totalmente conectada (*fuly connected*) se cada saída de uma camada é ligada a todo nó na camada subsequente.

Por outro lado, quando a saída de um nó for redirecionada como sua própria entrada ou de um outro nó localizado na mesma camada ou em uma precedente, a rede é dita ser do tipo *feedback*. Algumas redes desse tipo em especial que possuem alguma malha ou laço fechados são chamadas *recorrentes* (ou recursivas). Nesse caso, o processamento nodal não é exclusivamente definido pelos padrões de interconexão, mas também por uma fator temporal a ser levado em consideração. Essas configurações de rede também podem ser classificadas em *single-layer* ou *multi-layer* e podem ser visualizadas na Figura 3.8(c,d,e).

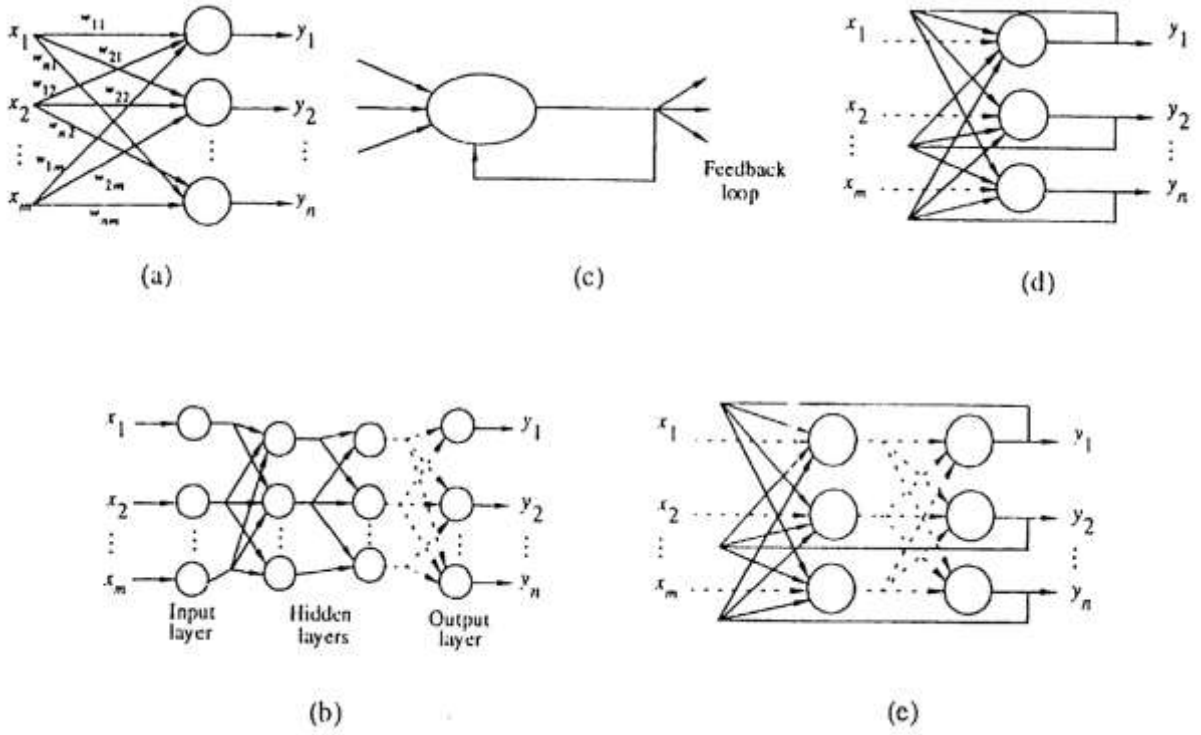


Figura 3.8 Geometrias básicas de conexão de redes. (a) *Single-layer feedforward network*. (b) *Multilayer feedforward network*. (c) Nó único com realimentação própria (*feedback unit*). (d) *Single-layer recurrent network*. (e) *Multilayer recurrent network*. (Lim, 1996)

Vale ressaltar que a escolha da topologia para uma rede neural é feita baseada em análises empíricas e dependente da experiência do projetista.

3.3 O Processo de Aprendizado das Redes Neurais

Redes neurais artificiais podem ser comparadas a uma caixa preta (*black box*), na medida em que, para uma certa entrada, produz uma saída desejada, mas o meio pela qual essa é gerada é implícito ao usuário, sendo tal fato alcançado devido ao processo de auto-organização da rede durante o período de aprendizado.

Um algoritmo de aprendizado é um método adaptativo, pelo qual a rede de unidades computacionais se auto-organiza a fim de implementar um comportamento desejado. No presente trabalho, isso será feito apresentando-se à rede algumas amostras do mapeamento entrada-saída objetivado. A correção de parâmetros sinápticos da rede é, então, efetuada iterativamente até que se consiga obter a resposta almejada. Esse processo é ilustrado na Figura 3.9 a seguir.

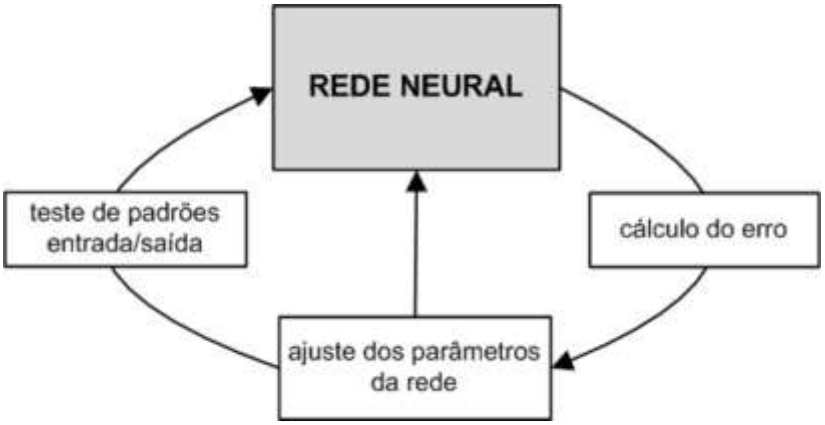


Figura 3.9 Esquemático do processo de aprendizado.

3.3.1 Classes de Algoritmos de Aprendizado

Os algoritmos de aprendizado podem ser classificados em dois tipos principais: aprendizado supervisionado (*supervised learning*) e aprendizado não-supervisionado (*unsupervised learning*).

Aprendizado supervisionado é aquele em que, a cada instante, quando uma entrada é aplicada à rede neural, a saída correspondente esperada é conhecida e fornecida. A resposta computada pela rede é, então, observada e comparada com aquela desejada e o erro entre elas

é medido. Esse erro computado pode ser usado para realizar o ajuste dos parâmetros nas conexões em cada camada, até se achar a melhor combinação de pesos que permita que ambas as respostas se aproximem umas às outras. Esse tipo de aprendizado é chamado *learning with a teacher* (ou *learning with error correction*). Um exemplo característico é o algoritmo *Backpropagation*, discutido no Apêndice 1.

Além disso, o erro pode ser usado para fornecer à rede a informação de que ela produz ou não a resposta esperada. Dessa maneira, há apenas um *bit* de informação realimentado indicando se a resposta está certa ou errada. Esse último tipo de aprendizado supervisionado é chamado *reinforcement learning*.

Aprendizado não-supervisionado é usado quando, para uma dada entrada, a resposta exata, a qual a rede deveria produzir, não é conhecida a princípio. A rede, por si só, deve ser capaz de descobrir os padrões, características, regularidades, correlações e categorias de cada entrada e associá-la a uma saída adequada. Um exemplo típico de aplicação desse algoritmo consiste na classificação de objetos sem o fornecimento de informações sobre as suas reais classes.

3.4 O Treinamento de uma Rede Neural

O treinamento supervisionado de uma rede neural é efetuado utilizando-se um conjunto de dados representativo e adequado, correspondentes a amostras de padrões, pares de entrada e saída desejada do sistema. A tarefa inicial de coleta desses dados requer uma análise cuidadosa sobre o problema a fim de minimizar ambiguidades e erros nos mesmos. Além disso, como os dados coletados devem ser significativos, há a necessidade de cobrir amplamente o domínio do sistema, incluindo as possíveis exceções e condições no seus limites, não devendo cobrir apenas as operações normais ou rotineiras. A partir desta base de dados, então, cabe aos algoritmos de aprendizado promover o treinamento.

Inicialmente a rede permanece inerte. Uma boa escolha dos valores iniciais dos pesos (e *biases*) da rede pode diminuir o tempo necessário para o treinamento. Normalmente, esses valores são números aleatórios uniformemente distribuídos, em um intervalo definido. O algoritmo de aprendizado modifica, então, individualmente os pesos das interconexões de tal forma que o aprendizado da rede reflita a ação desejada. Em outras palavras, a rede pode ajustar seus parâmetros até que se alcance o desempenho esperado de estimação dos dados.

Em virtude de a superfície de erro produzida pela rede ser não-linear, há então a possibilidade de que o processo de aprendizado direcione a matriz de pesos da rede para um ponto de mínimo local, que pode não corresponder aos valores mais apropriados aos propósitos de generalização de resultados. A tendência de convergência para um determinado ponto de mínimo fica então condicionada, principalmente, à posição espacial em que a matriz de pesos é iniciada, pois a grande maioria dos algoritmos de aprendizado é baseada em métodos de gradiente descendente. Para o exemplo da Figura 3.10, se a rede tivesse sua matriz de pesos iniciada em W^a, a tendência de convergência seria para o ponto de mínimo p^1; ao passo que se tivesse inciada em W^b, a tendência seria em direção a p^2.

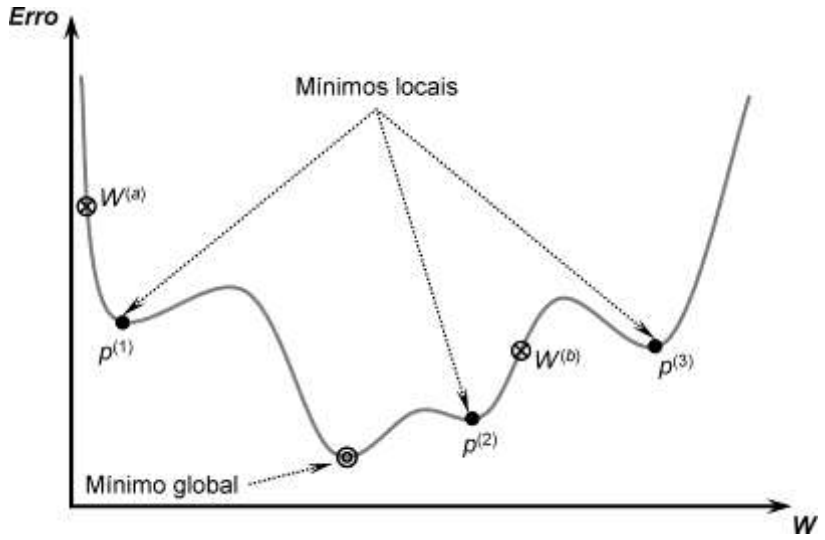


Figura 3.10 Pontos de mínimos locais e mínimo global associados à função erro. (Silva, 2010).

Visando evitar a convergência da rede para pontos de mínimos locais inapropriados, um dos procedimentos práticos consiste em executar o treinamento de cada estrutura de rede mais de uma vez, com diferentes matrizes de pesos iniciais (geradas aleatoriamente). Assim, dependendo de suas posições espaciais, a rede poderia convergir para pontos de mínimos locais, ou até mesmo globais, que possibilitariam uma melhor representação do comportamento do processo.

A atualização dos pesos pode ser feita baseada em dois modos: o *incremental training* (também denominado *on-line training*) e o *batch training* (ou *off-line training*). No *incremental training*, os pesos e *biases* são modificados cada vez que um novo padrão de entrada é apresentado à rede; já no *batch training*, pesos e *biases* são modificados apenas após todos os padrões de entrada serem apresentados à rede. O número de rodadas de apresentações de todos os padrões é chamado época.

No treinamento em lotes (*batch mode*), todos os padrões devem estar sempre disponíveis enquanto durar o processo de aprendizado. A aprendizagem padrão-a-padrão (*incremental mode*) é normalmente utilizada quando o comportamento do sistema a ser mapeado varia de forma bastante rápida, onde os padrões utilizados em determinado instante podem não mais representar o comportamento do processo nos instantes posteriores.

Quanto ao modo de treinamento, na prática, é mais utilizado o modo incremental devido ao menor armazenamento de dados, além de ser menos suscetível ao problema de mínimos locais da função de erro devido à pesquisa de natureza estocástica que realiza. Por outro lado, no modo *batch* se tem uma melhor estimativa do vetor gradiente do erro, o que torna o treinamento mais estável. A eficiência relativa aos dois modos de treinamento depende do problema, o qual está sendo tratado.

Quanto ao tempo de treinamento, vários fatores podem influenciar a sua duração, porém sempre será necessário utilizar algum critério de parada. Dentre alguns critérios, podemos citar, como exemplo, o número de épocas a ser limitado e pré-fixado com base em experiências anteriores ou análise empírica; o erro médio por ciclo[18]; e a capacidade de generalização[19] da rede. Em relação ao último citado, pode ocorrer que, em um determinado instante do treinamento, a generalização comece a degenerar, causando o problema de *overfitting* ou memorização, ou seja, a rede se especializa no conjunto de dados do treinamento, interpolando pontos desconhecidos erroneamente, e perde a capacidade de generalização. Nessa circunstância, o erro quadrático tende a ser baixo durante a fase de aprendizado e alto, durante a fase de testes.

A Figura 3.11 mostra um comportamento em que a rede está operando numa situação de *overfitting* (topologia 1), em contraste à rede (topologia 2) que está generalizando de forma bem satisfatória (Silva, 2010).

[18] O treinamento pára quando esse é inferior ao erro médio total do sistema.

[19] Generalização consiste na capacidade de uma rede neural em responder corretamente a uma entrada desconhecida, ou seja, distinta aos dados contidos na base para treinamento, interpolando a resposta a esses dados.

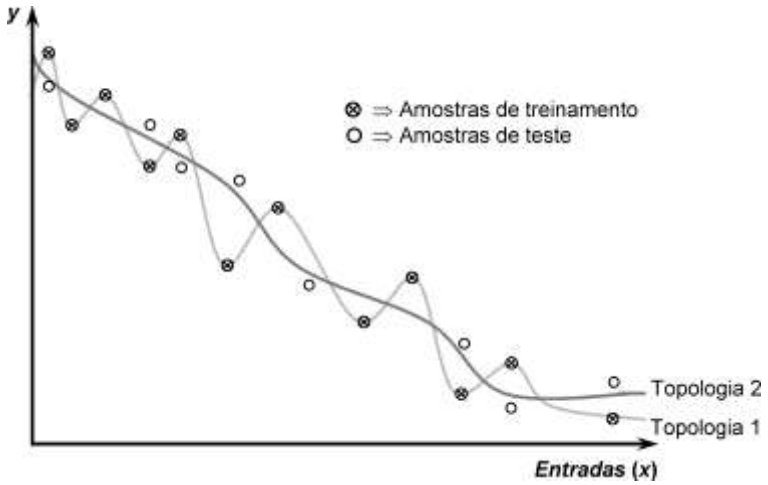


Figura 3.11 Comportamento de uma rede operando em situação de *overfitting* (topologia 1) e interpolando bem (topologia 2). (Silva, 2010).

Outro exemplo é mostrado na Figura 3.12, em que a rede, ao mapear uma função senoidal, está operando com (à esquerda) e sem (à direita) memorização excessiva.

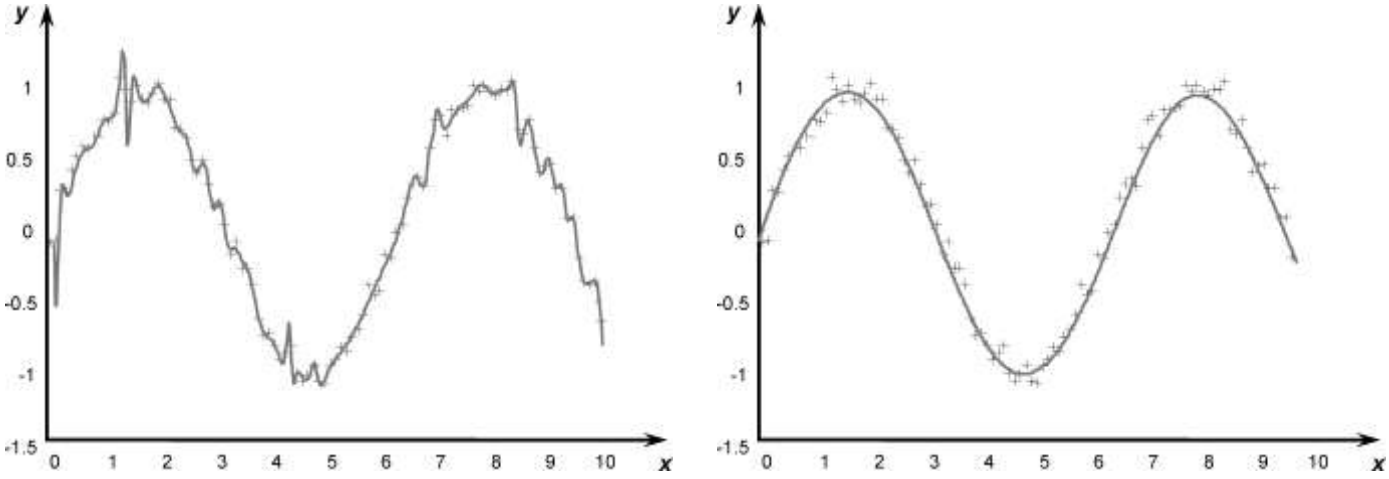


Figura 3.12 Comportamento de uma rede com *overfitting* (à esquerda) e sem *overfitting* (à direita). (Silva, 2010).

Em contrapartida, um número pequeno de épocas ou um erro alto desejado, durante o treinamento, pode ocasionar no não-aprendizado da rede, o que corresponde a uma situação de debilidade neural ou *underfitting*. Nesse caso, o erros quadrátricos tanto na fase de aprendizado como na de testes serão bem significativos.

O treinamento deve ser interrompido quando a rede apresentar uma boa capacidade de generalização e quando o número de épocas (*batch mode*) ou passos[20] (*incremental mode*) for

[20] Caso especial, referente ao modo incremental, de apresentação repetida do conjunto de dados de entrada, porém com atualização dos pesos a cada padrão.

o suficiente a permitir um erro apropriadamente pequeno, ou seja, menor que um limite admissível ou idealmente nulo. Assim, deve-se encontrar um ponto ótimo de parada com erro mínimo e capacidade de generalização máxima.

3.5 Fase de Validação

A fase de treinamento é seguida pela fase de validação, onde um conjunto de dados não utilizado previamente no treinamento de uma determinada rede é submetido à sua entrada a fim de avaliar o seu desempenho, ou seja, é verificado se a rede está suficientemente treinada ou não, além de medir sua capacidade de generalização. Após a rede ter aprendido o relacionamento entre as entradas e saídas que exprimem o comportamento do sistema, esta deve ser capaz de generalizar soluções, ou seja, a rede treinada deve ser capaz de produzir uma saída próxima daquela esperada a partir de quaisquer sinais inseridos em suas entradas. Tal habilidade difere radicalmente de técnicas tradicionais, porque uma estrutura de *software* como essa não depende de um conhecimento, à priori, de um programador para as possíveis soluções.

Em outras palavras, novos padrões de entrada são apresentados à rede, de modo a se esperar que ocorra uma interpolação dos resultados à base de dados treinados, ou seja, a rede deve ser capaz de reconhecer se novas entradas são similares aos padrões aprendidos para, então, produzir respostas similares.

3.6 Fase de Integração

Finalmente, com a rede neural treinada e validada, ela pode ser integrada em um sistema do ambiente operacional da aplicação. Para maior eficiência da solução, este sistema pode conter facilidades de utilização como interface conveniente e facilidades de aquisição de dados através de planilhas eletrônicas ou arquivos padronizados, por exemplo. A capacitação de usuários e uma boa documentação do sistema são necessários para o sucesso do mesmo. Além disso, o sistema deve periodicamente monitorar seu desempenho e fazer a manutenção da rede, quando necessário, ou indicar aos projetistas a necessidade de um novo treinamento.

3.7 Esquematização das Etapas do Projeto de uma Rede Neural por Diagrama de Blocos

A Figura 3.13 abaixo mostra um diagrama de blocos com as etapas descritas nas seções anteriores e necessárias para se desenvolver, treinar e validar uma rede neural em conjunto com um *software* simulador. A utilização de um pacote comercial, como o fornecido pelo Matlab® da *Mathworks*, pode auxiliar o desenvolvimento e treinamento de redes neurais. Entretanto, em algumas aplicações, o projetista que escolha ter maior controle sobre o projeto de uma rede ou embutir o código de tal rede em algum sistema de microprocessador específico, pode escrever o código em uma linguagem de alto nível adequada ou criar um mecanismo para "interfacear" os processos do projeto da rede, de modo a facilitar a interação com o usuário. Para sistemas que possuam modelo matemático, um simulador pode gerar os dados iniciais de treinamento, porém é desejável que a validação seja feita com dados reais da aplicação, para incluir todas as características do processo.

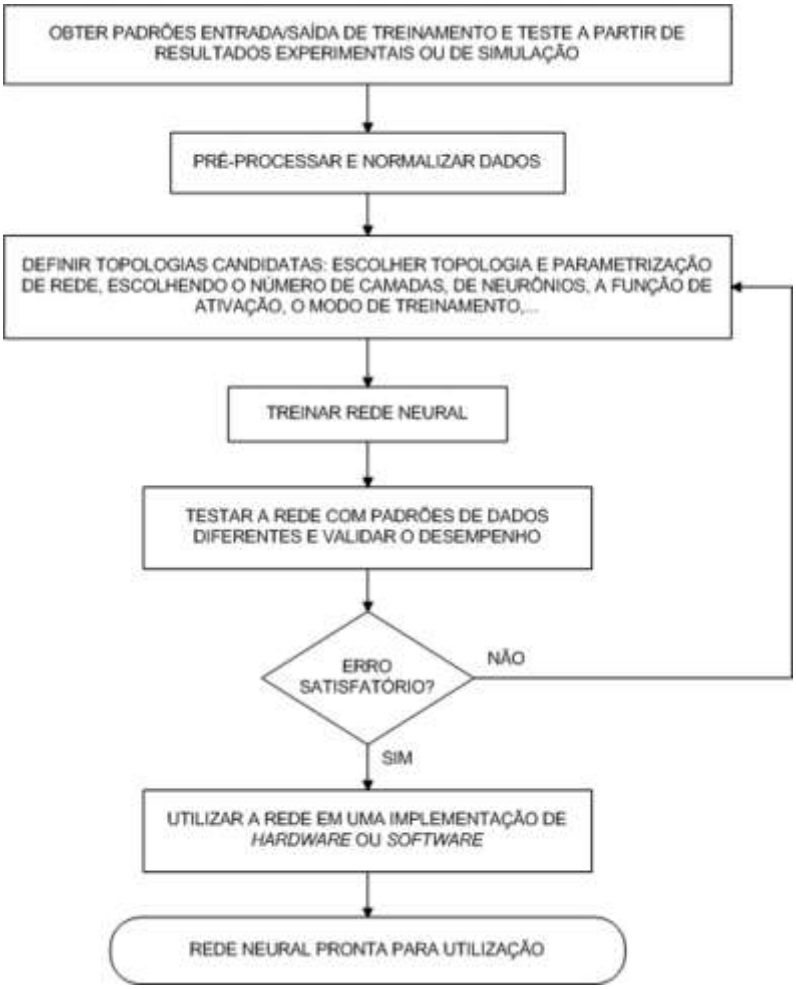


Figura 3.13 Fluxograma do projeto de desenvolvimento de uma rede neural.

3.8 Pontos Fortes e Fracos das Redes Neurais

Em projetos baseados em controle inteligente, as mais promissoras soluções são as que se utilizam e se baseiam nas características das redes neurais (e da lógica *fuzzy*). Conforme observado, as redes neurais oferecem características muito interessantes, destacadas a seguir:

- Adaptação por experiência: os parâmetros internos da rede, tipicamente seus pesos sinápticos, são ajustados a partir da apresentação sucessiva de exemplos (padrões, amostras, medidas) relacionados ao comportamento do processo, possibilitando a aquisição do conhecimento por experimentação;
- Capacidade de aprendizado: por intermédio da aplicação de um método de treinamento, a rede consegue extrair o relacionamento existente entre as diversas variáveis que compõem a aplicação;
- Habilidade de generalização por interpolação: após o treinamento da rede, essa é capaz de generalizar o conhecimento adquirido, possibilitando estimar soluções até então desconhecidas;
- Arquitetura distribuída, redundante e tolerante a falhas: uma característica intrínseca das redes neurais é a capacidade de tolerância a falhas. Em outro sistema de processamento, se uma parte falha, o sistema como um todo pode se deteriorar; enquanto que em uma rede neural a tolerância a falhas faz parte de sua arquitetura, devido à sua estrutura distribuída e redundante de processamento, ou seja, se um elemento neural falha, sua saída errônea é "sobrescrita" pelas saídas corretas de seus elementos vizinhos. Essa característica mostra que se pode extrair uma saída útil de um conjunto de dados imperfeito, incompleto ou com ruído. Além disso, representa a garantia da robustez da arquitetura frente a eventuais neurônios que se tornam inoperantes ou corrompidos;
- Facilidade de prototipagem: finalmente, como comentado anteriormente, a implementação da maioria das arquiteturas neurais pode, dependendo da especificidade da aplicação, ser facilmente prototipada em *hardware* ou em *software*, pois, após o treinamento, os seus resultados são normalmente obtidos por algumas operações matemáticas elementares.

Por outro lado, as neurais apresentam certos pontos fracos que devem ser considerados:

- Não é possível traçar a maneira pela qual uma rede neural chega a uma determinada solução. Nesse sentido, as redes neurais são semelhantes a muitos especialistas humanos que podem expressar uma opinião sem na realidade ter uma explicação plausível para tal feito;
- Uma vez que não se pode olhar para o sentido dos pesos e conexões de uma rede neural, torna-se impossível de se interpretar as causas de um comportamento particular; não é possível também de se mudar manualmente a estrutura de uma rede para se obter especificamente um comportamento desejado;
- Não existe um método de treinamento adequado a resolver todo e qualquer tipo de problema. Um mesmo método pode ser aplicado para a solução de inúmeros problemas distintos. A eficiência de um algoritmo de treinamento para um determinado sistema não é completamente compreendida, pois além de depender de um enfoque de tentativa e erro, não é conhecida, a princípio, sua relação com as regras de projeto a serem seguidas. Por exemplo, a definição do número de camadas e neurônios, ou o estabelecimento de uma determinada faixa de convergência depende de diversas experimentações e treinamentos com os dados;
- O tempo de treinamento não é previsível, podendo em alguns casos ser muito longo e além do esperado;
- O tempo de execução depende do número de conexões, sendo aproximadamente proporcional ao quadrado do número de neurônios utilizados. Isso significa que a adição de apenas alguns nós podem aumentar consideravelmente o tempo de execução.

3.9 Algumas Aplicações das Redes Neurais

Uma vez apresentada a descrição das redes neurais e como elas funcionam, nessa seção são citados exemplos práticos do uso dessas estruturas. Aplicações de redes neurais são inúmeras. As potenciais áreas de aplicabilidade podem ser enquadradas como segue.

a) Aproximador universal de funções: o objetivo consiste em mapear o relacionamento funcional entre variáveis reais de um sistema a partir de um conjunto conhecido de seus valores representativos. Técnicas de inteligência computacional são

preferencialmente utilizados em processos cuja modelagem por técnicas convencionais são de difícil obtenção;

b) Controle de processos industriais: o objetivo consiste em identificar ações de controle que permitam o alcance de requisitos de qualidade, eficiência e segurança do processo;
c) Identificação de processos;
d) Robótica, aeronaves, satélites;
e) Reconhecimento e classificação de padrões (reconhecimento de padrões em linhas de montagem; identificação de objetos tridimensionais; reconhecimento de voz, da face, de caracteres e formas, de impressões digitais; análise de imagens, vídeos e áudios; entre outros);
f) Agrupamento de dados (clusterização): identificação e detecção de similaridades e particularidades entre os diversos padrões de entrada a fim de possibilitar seu agrupamento;
g) Otimização de sistemas: o alvo consiste em minimizar ou maximizar uma função custo (objetivo), obedecendo a eventuais restrições impostas para o correto mapeamento do problema. Destacam-se problemas de otimização restrita, programação dinâmica e otimização combinatorial;
h) Área médica: atualmente a pesquisa está majoritariamente voltada para o modelamento de partes do corpo humano e para o reconhecimento de doenças. Como exemplo, redes neurais podem ser usadas experimentalmente para modelar o sistema cardiovascular humano. O diagnóstico é alcançado construindo um modelo de tal sistema e comparando-o com as medidas fisiológicas tomadas de um paciente. Caso essa rotina seja realizada regularmente, prejuízos potenciais podem ser detectados em estágios não avançados, facilitando o combate à doença.

   A seguir, é citado outro exemplo de como as redes neurais podem ser utilizadas como ferramentas de auxílio à tomada de decisões no diagnóstico médico, em geral. Durante o aprendizado neural, é submetida uma série de diagnósticos, de várias características, de pacientes com vários sintomas e os resultados de seus testes. Também são fornecidos os diagnósticos médicos para cada doença. Então, quando forem apresentados os dados de um novo paciente, com seus respectivos sintomas, a rede fornecerá um diagnóstico para os novos casos. Essa técnica essencialmente criará um sistema, o qual abrigará o conhecimento de vários médicos especialistas e fornecerá um diagnóstico inicial em tempo real a um determinado médico;

i) Por fim, uma área de destaque, com utilização ampla e crescente e que será abordada no presente trabalho é a de predição de sistemas: o objetivo, nesse caso, consiste em estimar valores futuros de um processo levando-se em consideração diversas medidas prévias observadas em seu domínio. Entre as aplicações existentes enquadram-se a análise e previsão de séries temporais; identificação de tendências e análise, pesquisa e previsão de mercados financeiros (atualmente, há um grande potencial do uso de redes neurais no que se refere a propostas de negócio, incluindo contabilidade, análise financeira, previsão de ações e identificação de fraudes); previsões climáticas; previsão em controle de processos (inclusive sistemas robóticos); entre outras.

Capítulo 4: Modelagem Analítica

Quando se fazem necessários a compreensão e o detalhamento de um processo, são empregadas representações do mesmo, as quais constituem os modelos. Estas abstrações têm como função primordial fornecer ao observador o nível de detalhamento necessário à compreensão do fenômeno (físico, químico, elétrico, mecânico, etc) a que o pesquisador se propôs observar.

Um modelo é uma representação simplificada da realidade ou uma interpretação de um fragmento de um sistema segundo uma estrutura de conceitos, ou seja, representa uma visão ou cenário de um fragmento do todo. Normalmente, para estudar um determinado fenômeno complexo são criados vários modelos, podendo esses ser representados por equações matemáticas (analíticos) ou por estruturas inteligentes como as redes neurais. A modelagem computacional utiliza, então, um conjunto de métodos, ferramentas ou formulações direcionados à solução de problemas complexos que envolvem um grande número de variáveis e/ou uma volumosa massa de dados.

O modelo analítico dos movimentos à ré de um RMMA é não-linear e muito complexo (Ferreira, 2004), o que tornam atrativas as técnicas de controle inteligente. Entretanto, ao se usar uma técnica inteligente, como as redes neurais, pode ser impraticável medir ou aprender um número muito grande de valores de entrada/saída, o que faz com que uma interpolação entre tais medidas seja sempre necessária, podendo, nesse caso, inserir erro no modelamento. Neste contexto, a modelagem analítica se revela interessante e pode ser obtida com suposições simplificadoras, mas que não comprometem o entendimento e a reprodução do sistema que se deseja modelar.

Embora tenha um papel importante na retratação do comportamento do sistema, o modelo analítico aqui equacionado destina-se a permitir a validação, via simulação, dos controladores neurais projetados e, principalmente, destina-se a permitir a geração dos modelos para as condições singulares de giro e para os ângulos críticos.

Nesse capítulo, portanto, é apresentado um modelo analítico original para as equações de movimento à ré de RMMA's, no espaço de configuração, desenvolvido por Ferreira (2011). Inicialmente, são explicitadas as equações analíticas dos movimentos à ré dos ângulos de configuração $(\theta_1, \theta_2, \dots, \theta_n)$ de um RMMA qualquer, com a arquitetura escolhida

anteriormente, em função do ângulo de direção das rodas dianteiras (γ) e da velocidade das rodas traseiras, no caso de tração traseira, ou da velocidade das rodas dianteiras, no caso de tração dianteira. Além disso, são apresentados os modelos das condições singulares de giro e dos ângulos críticos, extraídos do modelo analítico mencionado. Posteriormente, os modelos são validados para o RMMA utilizado, em condições reais de operação.

4.1 Modelo Analítico Discreto no Espaço de Configuração

O modelo analítico correspondente foi desenvolvido supondo pequenos movimentos planos para trás, com velocidade constante em cada intervalo, decompostos em rotações e translações elementares desacopladas.

Em Ferreira (2002), um primeiro modelo foi deduzido segundo uma sistemática mais complexa baseada na suposição de um único centro ou raio de giro da composição *truck-trailers*, com a cadeia articulada somente em configuração convexa (todas as variáveis de configuração com mesmo sinal), restrita à ligação *on-axle* entre o *truck* e o primeiro *trailer* e somente tração dianteira. O novo modelo de dedução mais simples e mais geral que o anterior, foi deduzido para ligações *on-axle* ou *off-axle*, com tração no *truck* dianteira ou traseira, para configurações convexas ou côncavas (nesse último, alguma variável de configuração possui sinal diferente das outras). Os dois modelos desprezam a maior parte dos efeitos dinâmicos e perturbações e são analiticamente equivalentes dentro das restrições do primeiro.

O primeiro modelo, em Ferreira (2002), foi utilizado em trabalhos anteriores como núcleo do simulador GMSARA (Kulitz, 2003) para desenvolvimento de estratégias de controle, para análise de navegação e para estabelecimento das restrições de ângulos críticos, em manobras de RMMA's. Este simulador também foi utilizado para estudar diversas alternativas de modelagem *fuzzy*, inclusive as técnicas de clusterização para o desenvolvimento, em simulação, de preditores *fuzzy*, para navegação e manobras assistidas de RMMA's (Ferreira, 2004). Além disso, este simulador foi utilizado por Silva (2003) para gerar dados para o estabelecimento de um controlador neural inverso reduzido, baseado em rede estática, para um robô virtual de três graus de liberdade.

O novo modelo proposto, por sua vez, fornecerá subsídios a trabalhos futuros, pois suas equações podem vir a fazer parte de um trabalho mais amplo de desenvolvimento de

outro simulador mais complexo para analisar RMMA's incorporando restrições tanto no espaço de configurações quanto no de tarefas.

Como se sabe, um preditor completo é aquele que considera a velocidade como uma informação adicional na obtenção/predição dos ângulos de configuração ou de suas variações. No modelo analítico, essa variável aparece explícita no seu equacionamento, diferentemente do preditor neural. Nesse caso, seu valor pode ser medido experimentalmente ou obtido analiticamente após a coleta de algumas trajetórias descritas pelo RMMA.

As equações do modelo discreto podem ser implementadas para um período de amostragem T correspondendo ao horizonte de predição H do preditor e supondo a velocidade, V^k, constante no período de amostragem. A Equação 4.1 explicita o cálculo das amostras dos ângulos de configuração θ_{r+1}^{k+h}, no instante (*k*+*h*)T, sendo *h* = H/20, para *r* = 1,2,...,(*n*-1), onde *n* é a dimensão do vetor de configuração.

$$\theta_{r+1}^{k+h} = \theta_{r+1}^{k} + \mathrm{H}\, v_r^k \left[\left(1 + \left(\frac{B_{r+1}}{A_{r+2}} \right) \cos\left(\theta_{r+1}^k \right) \right) \frac{sen(-\theta_r^k)}{A_{r+1}} + \frac{cos(\theta_r^k)\, sen\left(\theta_{r+1}^k \right)}{A_{r+2}} \right] \qquad (4.1)$$

Somente o resultado final do cálculo dos ângulos de configuração é apresentado, devido à complexa dedução matemática do modelo. Na Equação 4.1, acima, para *r* = 2,...,(*n*-1), tem-se:

$$v_r^k = v_{r-1}^k cos\left(\theta_{r-1}^k \right), \text{ sendo} \qquad (4.2)$$

$$\begin{cases} v_1^k = |V^k| cos(\gamma^k), & \text{para tração dianteira} \\ v_1^k = |V^k|, & \text{para tração traseira} \end{cases}$$

A Equação 4.3 explicita o cálculo das amostras do ângulo de configuração θ_1^{k+h}, para tração dianteira no *truck*.

$$\theta_1^{k+h} = \theta_1^k + \mathrm{H}\, |V^k| \left[\left(1 + \left(\frac{B_1}{A_2} \right) \cos\left(\theta_1^k \right) \right) \frac{sen(\gamma^k)}{A_1} + \frac{cos(\gamma^k) sen\left(\theta_1^k \right)}{A_2} \right] \qquad (4.3)$$

Por fim, a Equação 4.4 explicita o cálculo das amostras do ângulo de configuração θ_1^{k+h}, para tração traseira no *truck*.

$$\theta_1^{k+h} = \theta_1^k + \mathrm{H}\, |V^k| \left[\left(1 + \left(\frac{B_1}{A_2} \right) \cos\left(\theta_1^k \right) \right) \frac{cos(\gamma^k) sen(\gamma^k)}{A_1} + \frac{sen\left(\theta_1^k \right)}{A_2} \right] \qquad (4.4)$$

Para o protótipo de RMMA em questão, o modelo é restrito a três graus de liberdade, tração traseira e engate *trailer-trailer* com ligação *off-axle* e *truck-trailer*, *on-axle*. Assim, as

equações que caracterizam o preditor analítico de horizonte H são a Equação 4.4 e a Equação 4.5, sendo essa última extraída da Equação 4.1 ao se atribuir *r* = 1.

$$\theta_2^{k+h} = \theta_2^k + \mathrm{H}\,|V^k| \left[\left(1 + \left(\frac{B_2}{A_3} \right) \cos\left(\theta_2^k \right) \right) \frac{sen\left(-\theta_1^k \right)}{A_2} + \frac{cos\left(\theta_1^k \right) sen\left(\theta_2^k \right)}{A_3} \right] \tag{4.5}$$

Os parâmetros geométricos (A_i, B_i) para o protótipo utilizado se encontram na Figura 2.6. Os valores atribuídos a esses devem estar no Sistema Internacional (S.I.) de medida.

Como se percebe, a velocidade de acionamento do *truck*, V, está explicita como função das variáveis de configuração nas equações de derivadas (modelo contínuo) ou de variações (modelo discreto).

$$\delta\theta_i^{k+h} = |V_i| f(\theta) \tag{4.6}$$

Logo, essa velocidade pode ser estimada, via técnicas convencionais, a partir de dados de variação, $\delta\theta_i$, de qualquer variável de configuração e a partir da configuração global, θ, que intervêm pela função não-linear, $f(\theta)$, de senos e cossenos das diversas variáveis de configuração θ_j, j = 1,...,n. Assim:

$$|V_i| = \frac{\delta\theta_i^{k+h}}{f(\theta)} \tag{4.7}$$

No caso do protótipo da presente aplicação, esse equacionamento é aplicado a todas as configurações (ou a todos os pares de ângulos das articulações) em cada trajetória e a média dos valores obtidos é calculada. Por fim, a média pode ser novamente calculada, considerando todas as trajetórias. A estimação da velocidade se torna redundante ao se usar variações de mais de uma variável de configuração, mas deve ser efetivada na prática face ao grande número de ruídos e perturbações presentes no sistema.

Outra maneira bem simples e prática de se estimar a velocidade se dá experimentalmente fazendo-se uso do próprio protótipo. Para uma distância pré-fixada, é medido o tempo gasto pelo RMMA até se completar o percurso. Esse procedimento é repetido algumas vezes e, por fim, o tempo considerado consiste na média daqueles medidos. Essa abordagem foi adotada tendo em vista a necessidade de uma implementação ainda mais refinada do controle de velocidade para o protótipo, o que inviabilizou a obtenção de resultados satisfatórios na abordagem anterior. A velocidade obtida foi de 0.1667*m/s*.

4.2 Modelo das Condições Singulares de Giro

Mesmo no caso da síntese baseada no uso de técnicas de inteligência computacional, os modelos analíticos não podem ser descartados. A expressão analítica das singularidades é extremamente importante para o desenvolvimento de diversas estratégias de controle e predição. A síntese de controladores, via aproximadores numéricos, do mesmo modo que a de preditores, necessita dos modelos das condições singulares de giro para a geração dos dados complementares ao treinamento das redes neurais ou para a geração de regras *fuzzy*. A razão disso é que sendo o giro da cadeia mecânica articulada uma condição singular de equilíbrio instável, não se consegue obter dados suficientes em movimentos em malha aberta sobre os giros e sim somente dados sobre as transições entre giros. Quase a totalidade dos dados reais é representativa somente em relação às transições entre giros, não provendo nenhuma informação sobre alguma das infinitas configurações de giro.

A utilização de algum controlador em malha fechada, para a obtenção dos dados de giro, deve polarizar os dados e limitar o desempenho do controlador resultante ao seu desempenho. Portanto, ensaios sobre a utilização de dados em malha fechada para identificação de sistemas ou síntese de controle não se revelam frutíferas.

O giro se dá segundo configurações convexas e constantes, para um valor do ângulo de direção constante. Como mencionado, uma configuração convexa é tal que todos os ângulos de configuração têm o mesmo sinal. Numa situação de giro, o ângulo de direção, γ, e os de configuração, θ_i, devem permanecer constantes durante o movimento. Analiticamente, dado o valor desejado do último ângulo de configuração, θ_n, os sucessivos ângulos de configuração e o ângulo da direção são calculados, no sentido reverso da cadeia, pelas Equações 4.8, 4.9 e 4.10, abaixo. Essas equações foram obtidas diretamente a partir do novo modelo, descrito anteriormente, considerando a derivada nula ou os ângulos invariantes no movimento. Nessas, os ângulos a serem calculados dependem dos parâmetros geométricos longitudinais do i-ésimo elemento da cadeia do RMMA, A_i e B_i, ilustrados na Figura 2.4 e descritos no Capítulo 2. Assim, a partir da última configuração, θ_n, calculamos sucessivamente os outros ângulos de configuração:

$$\theta_k = \left| tan^{-1}\left(\frac{A_{k+1}\, sen(\theta_{k+1})}{A_{k+2} + B_{k+1}\, cos(\theta_{k+1})}\right)\right| \left(sign(\theta_{k+1})\right) \qquad (4.8)$$

A Equação 4.8 fornece θ_k em função de θ_{k+1}, para k = 1,...,(n-1). Já, o cálculo do ângulo de direção depende do tipo de tração do RMMA.

Para tração traseira:

$$\gamma = \frac{1}{2}\left|sen^{-1}\left(\frac{2\,A_1 sen(\theta_1)}{A_2 + B_1\,cos(\theta_1)}\right)\right|\left(-sign(\theta_2)\right) \tag{4.9}$$

Para tração dianteira:

$$\gamma = \left|tan^{-1}\left(\frac{A_1 sen(\theta_1)}{A_2 + B_1 cos(\theta_1)}\right)\right|\left(-sign(\theta_2)\right) \tag{4.10}$$

Para o RMMA em questão, com tração traseira, dado θ_2, obtém-se:

$$\theta_1 = \left|tan^{-1}\left(\frac{A_2\,sen(\theta_2)}{A_3 + B_2\,cos(\theta_2)}\right)\right|\left(sign(\theta_2)\right) \tag{4.11}$$

$$\gamma = \frac{1}{2}\left|sen^{-1}\left(\frac{2\,A_1 sen(\theta_1)}{A_2}\right)\right|\left(-sign(\theta_2)\,\right) \tag{4.12}$$

Por fim, a Tabela 4.1 explicita alguns resultados para as condições de giro no sentido negativo de θ_{2g}. Os resultados correspondentes no sentido positivo de θ_{2g} são simétricos e, portanto, não serão mostrados. O Apêndice 1 mostra todas as condições singulares calculadas, de onde foram extraídos os elementos da tabela abaixo. Essas singularidades irão fazer parte das massas de dados das redes projetadas.

Tabela 4.1 Exemplos de condições singulares de giro.

$\boldsymbol{\theta_{2g}}$ (°)	$\boldsymbol{\theta_{1g}}$ (°)	$\boldsymbol{\gamma_g}$ (°)
-54	*-31.607107*	*19.719728 = 20*
-48	*-28.685078*	*17.789051 = 18*
-42	*-25.579842*	*15.778807 = 16*
-36	*-22.300820*	*13.692357 = 14*
-30	*-18.861011*	*11.534728 = 12*
-24	*-15.277561*	*9.312965 = 9*
-18	*-11.572044*	*7.036308 = 7*
-12	*-7.770356*	*4.716158 = 5*
-6	*-3.902177*	*2.365816 = 2*
0	*0.000000*	*0.000000 = 0*

4.3 Modelo dos Ângulos Críticos

As expressões analíticas dos ângulos críticos, para uma dada geometria de RMMA e para certo limite do ângulo de direção, são muito úteis para analisar as possibilidades de

manobra num certo ambiente restrito e servem também, no projeto de RMMA's, para definir, à priori, as especificações construtivas de parâmetros e restrições sobre a direção, de modo que o RMMA atenda aos requisitos de manobrabilidade impostos pelas tarefas para a atuação em uma área restrita. Além disso, o modelo para os ângulos críticos é usado não só para definir o domínio dos ângulos antes de se atingir o *jackknife*, mas também é essencial para a predição das configurações críticas, conforme se percebe na Figura 1.3.

Para um dado RMMA, os ângulos críticos dependem basicamente dos parâmetros geométricos da cadeia mecânica articulada e dos limites físicos do ângulo da direção (no caso do protótipo em questão esses limites são de ±20°). A expressão analítica dos ângulos críticos deve fazer parte de um sistema supervisório para controle e predição em navegação e manobras de RMMA's. Os correspondentes valores críticos dos ângulos de configuração, para um dado valor máximo do ângulo de direção, podem ser calculados recursivamente, no sentido direto da cadeia articulada, segundo a Equação 4.13, deduzida a partir do modelo analítico proposto e, mais especificamente, a partir da Equação 4.8 de cálculo dos ângulos de giro.

$$tan(\theta_k) = \left(\frac{A_{k+1}\, sen(\theta_{k+1})}{A_{k+2} + B_{k+1}\, cos(\theta_{k+1})}\right) \rightarrow A_{k+2}\, tan(\theta_k) + B_{k+1}\, cos(\theta_{k+1}) tan(\theta_k) = A_{k+1}\, sen(\theta_{k+1})$$

$$\rightarrow A_{k+1}\, sen(\theta_{k+1}) - B_{k+1} tan(\theta_k) \cos(\theta_{k+1}) = A_{k+2}\, tan(\theta_k)$$

$$\rightarrow \left(\frac{A_{k+1}}{A_{k+2}\, tan(\theta_k)}\right)\, sen(\theta_{k+1}) + \left(\frac{-B_{k+1}}{A_{k+2}}\right) \cos(\theta_{k+1}) = 1 \qquad (4.13)$$

As restrições decorrentes podem ser reduzidas a uma equação similar da forma $A\, sen(X) + B\, cos(X) = 1$, cuja solução, para $X = \theta_{k+1}$, é:

$$\theta_{k+1} = X = sen^{-1}\left(\frac{1}{\sqrt{A^2 + B^2}}\right) - sen^{-1}\left(\frac{B}{\sqrt{A^2 + B^2}}\right) \qquad (4.14)$$

onde, para k = 0,1,...,(n-1), tem-se:

$$B = -\frac{B_{k+1}}{A_{k+2}} \qquad (4.15)$$

para k = 0, tem-se:

$$\begin{cases} A = -\left(\frac{A_1}{A_2}\right)\frac{1}{tan(\gamma)}, & \text{para tração dianteira} \\ A = -\left(\frac{A_1}{A_2}\right)\frac{1}{sen(\gamma)cos(\gamma)}, & \text{para tração traseira} \end{cases} \qquad (4.16)$$

e para k = 1,...,(n-1), tem-se:

$$A = \left(\frac{A_{k+1}}{A_{k+2}}\right)\frac{1}{tan(\theta_k)} \tag{4.17}$$

Para o caso da aplicação do presente trabalho, tem-se, então:

$$\theta_1 = \theta_2 = arcsen\left(\frac{1}{\sqrt{A^2 + B^2}}\right) - arcsen\left(\frac{B}{\sqrt{A^2 + B^2}}\right) \tag{4.18}$$

Para θ_1:	Para θ_2:
$B = -\frac{B_1}{A_2}$ $A = -\left(\frac{A_1}{A_2}\right)\frac{1}{sen(\gamma)cos(\gamma)}$,	$B = -\frac{B_2}{A_3}$ $A = \left(\frac{A_2}{A_3}\right)\frac{1}{tan(\theta_1)}$

O protótipo em questão apresenta uma restrição muito importante no ângulo da direção. Esta restrição poderia ser relaxada por modificação de *hardware* mecânico, possibilitando uma maior manobrabilidade. A Tabela 4.2 ilustra o cálculo dos ângulos críticos para os parâmetros geométricos desse protótipo e para diferentes limites do ângulo de direção. Em outras palavras, a tabela mostra os limites geométricos dos ângulos de configuração. Esses ângulos têm também um limite físico inerente ao RMMA, em geral apreciavelmente menor que 90º, porém, para efeitos de análise, a restrição física da configuração será considerada como aproximadamente 90º. São apresentados alguns resultados para tração traseira e para tração dianteira.

Tabela 4.2 Ângulos críticos para alguns ângulos de direção.

Tração Traseira			***Tração Dianteira***		
$\lvert\boldsymbol{\gamma_c}$ (°)$\rvert$	$\lvert\boldsymbol{\theta_{1c}}$ (°)$\rvert$	$\lvert\boldsymbol{\theta_{2c}}$ (°)$\rvert$	$\lvert\boldsymbol{\gamma_c}$ (°)$\rvert$	$\lvert\boldsymbol{\theta_{1c}}$ (°)$\rvert$	$\lvert\boldsymbol{\theta_{2c}}$ (°)$\rvert$
20	*32.0257*	*54.8926*	**20**	*36.9094*	*66.0798*
22	*34.9662*	*61.4399*	**22**	*41.8085*	*79.3573*
25	*39.1968*	*71.9479*	**25**	*50.3007*	*123.1440*
27	*41.8697*	*79.5423*	**27**	*57.2159*	*134.4080*
30	*45.5999*	*92.3031*	**30**	*72.2937*	*160.8718*
31	*46.7551*	*97.1695*	**31**	*82.4891*	*179.8721*
33	*48.9097*	*109.3027*	**33**	*92.5424*	*188.9624*
36	*51.6857*	*125.3651*	**36**	*96.7684*	*181.2492*
38	*53.1773*	*127.7751*	**38**	*99.5653*	*176.0425*
40	*54.3376*	*129.6638*			
42	*55.1329*	*130.9660*			
43	*55.3853*	*131.3805*			
46	55.5376	*131.6310*			

A partir desses fatos, algumas conclusões muito interessantes podem ser extraídas da tabela:

i. A tração dianteira possibilita maior manobrabilidade que a tração traseira;
ii. A restrição geométrica sobre θ_2 deixa de existir para valores de γ a partir de 30º para tração traseira, e a partir de aproximadamente 23º para tração dianteira;
iii. A restrição geométrica sobre θ_1 é sempre maior do que sobre θ_2;
iv. Na tração traseira, o efeito no ângulo θ_1 da restrição geométrica sobre γ é crítico, sendo que o limite de γ até onde é possível melhorar essa restrição é de aproximadamente 40º (a partir desse ponto, os valores críticos de θ_1 praticamente não se alteram);
v. A Tabela 4.2 mostra que não adianta modificar o *hardware* do protótipo para aumentar o limite de γ para além dos 40º;
vi. A Tabela 4.2 mostra também que se a tração fosse dianteira bastaria aumentar este limite para 32º (a partir desse ponto, porém, os valores críticos de θ_1 são todos iguais à restrição física de 90°).

O próximo capítulo discute uma das partes práticas realizadas no decorrer do presente trabalho, referente ao modelamento neural da planta para o projeto de preditores completos a horizonte fixo.

Capítulo 5: O Problema de Predição: Modelagem Neural do Robô Móvel Multiarticulado

A literatura demonstra um grande progresso, nas últimas décadas, nas técnicas de controle não-linear. Contudo, para os problemas de identificação e modelagem de sistemas fortemente não-lineares e de plantas bem complexas ou desconhecidas, a lógica nebulosa (*fuzzy*) e as redes neurais têm se apresentado como ferramentas eficientes, gerando importantes contribuições no campo da modelagem e identificação desses tipos de sistemas.

Uma vez que as redes neurais e os sistemas *fuzzy* são ambos por natureza estimadores numéricos livres da necessidade de modelos, dividem a habilidade comum de melhorar a inteligência de sistemas, trabalhando em ambientes de incertezas, imprecisão e ruídos.

Ao longo deste capítulo será discutido o emprego das redes neurais para obtenção de preditores completos, a horizonte fixo, do RMMA. São detalhados os procedimentos da *Interface* implementada, desde a aquisição e pré-processamento de dados, passando pela montagem das bases para treinamento de diversas redes neurais até a validação das mesmas como preditores ou controladores.

5.1 A Proposta de Modelagem

Na literatura, vários autores propõem estratégias de síntese, principalmente *fuzzy*, baseadas em informações puramente geométricas (somente configuração $\underline{\theta}$), a cada instante de amostragem, sem utilizar nenhuma informação sobre a velocidade ou variações. O uso de informações da velocidade vai melhorar o desempenho das soluções, mas implica em adicionar mais entradas aos aproximadores numéricos. Considerar as variações dos ângulos de configuração, $\delta\theta_{1k}$ e $\delta\theta_{2k}$, como entradas adicionais de cada rede neural inclui implicitamente informação sobre a velocidade.

Na abordagem neural, computacionalmente muito eficiente, o uso de mais entradas não introduz demandas significativas no processo de síntese e de utilização. A abordagem *fuzzy*, ao contrário, a cada nova variável de entrada, tem o número de regras multiplicado pelo número de partições da nova variável; na prática, isso vai limitar o uso de variações da

configuração a uma variação com cinco partições ou a duas variações com três partições cada. Esse fato motiva o uso da abordagem neural no projeto de preditores e controladores completos. Um preditor completo é definido, portanto, como o mapeamento:

$$\left(\underline{\gamma}, \underline{\theta}, \delta\underline{\theta}\right)_k — \left(\delta\underline{\theta} \; ou \; \underline{\theta}\right)_{k+h} \tag{5.1}$$

Outra vantagem de se utilizar a solução neural é que ela utiliza dados indistintamente gerados por modelos ou medidos do sistema real (Ferreira, 2002)[21]. Ao se utilizar dados do sistema real, situações tipo derrapagens, deslizamentos, folgas, saturações, elasticidades, ruídos e diversas outras perturbações estariam embutidos nos dados e seriam representados no modelo aproximado.

5.2 Coleta de Dados

5.2.1 A Estratégia de Obtenção dos Dados para o Treinamento

Na abordagem do problema, a primeira tarefa é a coleta de dados sobre o protótipo. Inicialmente, é necessário estabelecer uma estratégia adequada e viável na prática para a obtenção da massa de dados a ser usada para o treinamento e validação de uma rede. Esta preocupação ampara-se na experiência revelada pela literatura, onde uma aplicação bem sucedida de rede neural requer uma base de dados dita apropriada. Uma base de dados apropriada deve conter informações relevantes e representativas do sistema a ser modelado, buscando varrer o universo de configurações possíveis do RMMA de modo abrangente, permitindo à rede apreender a dinâmica que se deseja modelar e generalizar o conhecimento.

A questão chave concentrou-se, portanto, em como percorrer o domínio do ângulo da direção e dos ângulos de configuração sem, no entanto, perder informações importantes ao treinamento das redes. Um cuidado tomado na definição da solução para a coleta de dados foi o de retratar o conhecimento do especialista sobre a dinâmica da planta. Assim, o espaço de configuração foi discretizado adequadamente, para cada variável, em valores iniciais, dentro das restrições dos ângulos críticos. As configurações iniciais foram obtidas pela combinação das variáveis discretizadas. Neste processo, escolheu-se um mesmo sinal para todas as variáveis de configuração, de modo a manter uma convexidade. Para cada configuração

[21] Observa-se que para a medição dos dados do sistema real, os ângulos das rodas dianteiras e os das articulações do RMMA devem ser instrumentados, conforme discutido no Capítulo 2.

inicial foi associada uma discretização do domínio do ângulo de direção, o que forneceu a combinação final de condições iniciais $\{(\gamma, \theta_1, \theta_2)_i\}$ dos experimentos. O número final de movimentos deveria ser suficientemente grande para ser representativo, sem tornar extenuante a tarefa de coleta de dados. A solução empregada consistiu em fracionar o domínio de cada uma das variáveis (θ_1 e θ_2) em cinco partições assumindo os valores {0°, ±1°, ±3°, ±7°, ±11°} e o do ângulo da direção (γ) nos seguintes valores {0°, ±1°, ±2°, ±3°, ±5°, ±7°, ±10°, ±13°, ±16°, ±20°}. Assim, foram obtidas 475 condições iniciais convexas implementáveis de movimento.

O processo de coleta de dados, portanto, transcorreu efetuando-se a gravação dos valores das variáveis de configuração, amostrados a um tempo fixo T de 20*ms*, ao longo do movimento, para o veículo partindo de cada uma das condições iniciais até o mesmo atingir uma situação de *jackknife*. A determinação do tempo de amostragem, T, se deu experimentalmente após inúmeras análises. O valor escolhido foi suficiente para não permitir a perda de dados durante amostragens (leituras) consecutivas, ao longo de uma trajetória (o que ocorreria com um valor relativamente grande de T) e suficiente a não permitir a leitura de valores repetitivos de um mesmo ângulo em trechos de trajetórias onde o seu valor muda (o que ocorreria com um valor relativamente pequeno de T).

Durante a presente etapa, não houve preocupação em seguir ou observar uma trajetória específica para o veículo; ao contrário, se fez imprescindível amostrar os mais diversos comportamentos, dadas certas condições iniciais da cadeia cinemática, a fim de fornecer aos modelos neurais subsídios suficientes ao aprendizado do comportamento do RMMA em malha aberta.

Como o sistema é simétrico, foram realizados 475 experimentos com o veículo se movendo, a partir de configurações convexas, em uma única direção[22], além do movimento alinhado. O conjunto de dados dos 474 movimentos simétricos aos realizados pôde ser obtido a partir daqueles coletados, trocando-se os sinais de cada ângulo.

Foi utilizado o próprio *Hyperterminal* do sistema operacional Windows para "interfacear" tanto a geração de comandos e a aquisição de dados quanto a comunicação do computador com o *software* embarcado no protótipo. Os comandos necessários à realização de trajetórias por parte do protótipo são constituídos por caracteres ASCII (necessariamente minúsculos), os quais identificam as funções implementadas. Todos os comandos

[22] No sentido horário (negativo) ou anti-horário (positivo) de rotação de θ_2. Foi escolhido o sentido horário, devido às melhores condições físicas do piso nessa direção.

implementados se encontram em Barreto (2010) e os utilizados nos experimentos são mostrados a seguir:

- **n**: seleciona velocidade zero para o protótipo, desliga o motor;
- **c**: reinicializa todo o sistema embarcado;
- **w**: controle manual de velocidade: aumenta a velocidade em uma unidade;
- **s**: controle manual de velocidade: diminui a velocidade em uma unidade;
- **o<sinal><valor>**: Controle automático de velocidade. Caso **<sinal> = –,** o movimento é para frente; caso **<sinal> = +**, o movimento é para trás. O parâmetro **<valor>** é um inteiro de 0 a 9 e indica a velocidade;
- **v<sinal><valor>**: Define o ângulo da direção das rodas. Caso **<sinal>= +** o ângulo é à esquerda (sentido anti-horário); caso **<sinal> = –**, o ângulo é à direita (sentido horário). O parâmetro **<valor>** é um inteiro de dois dígitos variando de 00 a 20 e indica o valor absoluto do ângulo;
- **a**: varia o ângulo da direção em +1°, no sentido anti-horário;
- **d**: varia o ângulo da direção em -1°, no sentido horário;
- **z**: requisita os valores lidos dos potenciômetros;
- **i<valor>**: O *hardware* inicia a gravação dos ângulos das articulações e do valor do *encoder* (contador de 0 a 255). Além disso, "*reseta*" os valores lidos no experimento anterior. O parâmetro **<valor>** é um inteiro de 01 a 99 e indica o fator multiplicativo que gere um múltiplo de 4*ms*. Esse valor múltiplo corresponderá ao intervalo de tempo entre a gravação dos ângulos (ou tempo de amostragem, T). Foi escolhido, para a coleta de dados, **<valor>** = 5, de modo a permitir um tempo de amostragem de 20*ms* ($5 \times 4ms = 20ms$);
- **r**: requisita os valores dos ângulos de configuração salvos no *hardware* embarcado.

5.2.2 A Etapa de Pré-Processamento

Uma vez terminada a coleta, foi implementada a etapa de pré-processamento, onde foram feitas manipulações sobre os dados de modo a colocá-los numa forma adequada ao treinamento das redes.

a) Conformação dos dados: Os valores lidos pelos potenciômetros do protótipo devem ser transformados para ângulos em graus, definindo θ_1 e θ_2.

Uma vez que não foi utilizado no protótipo nenhum tipo de limitador mecânico de rotação para os potenciômetros, vários valores de resistência são permitidos.

Inicialmente, é necessário a determinação do valor correspondente ao ângulo zero de cada variável de configuração a cada nova montagem dos elementos da composição nos engates ou periodicamente após a realização de alguns experimentos com o robô. O valor do ângulo zero, para cada junta, é obtido quando se alinha os elementos da composição. A partir desse ponto, os demais ângulos podem ser obtidos.

Uma variação angular gera uma diferença de potencial no circuito, ao qual o potenciômetro está acoplado, sendo essa possível de ser lida por meio de um conversor A/D de 10 *bits*. Assim, sabendo-se que uma variação de 1024 unidades na escala do conversor representa uma variação angular de 360°, o que faz com que a mínima variação angular detectável (possível de ser lida pelos potenciômetros) seja de 0.352°, os valores dos ângulos de configuração podem ser determinados por uma simples regra de três, como segue:

$$\begin{cases} 1024 & = 360° \\ V_{lido_{pot(\theta_i)}} - V_{alinhado_{pot(\theta_0)}} & = \theta_i \end{cases}$$

$$\theta_i = \left(V_{lido_{pot(\theta_i)}} - V_{alinhado_{pot(\theta_0)}}\right)\left(\frac{360}{1024}\right), \quad i = 1,2 \tag{5.2}$$

onde $V_{lido_{pot(\theta_i)}}$ corresponde ao valor lido do potenciômetro, codificado pelo conversor A/D, enquanto que $V_{alinhado_{pot(\theta_0)}}$ designa o valor lido quando o veículo se encontra alinhado.

Esse cálculo é realizado no computador remoto, permitindo, assim, uma maior precisão do resultado e, ao mesmo tempo, diminuindo o processamento no *hardware* embarcado.

b) Filtragem dos dados: Foram considerados somente os dados a partir de uma faixa de variação de pelo menos um dos valores lidos dos potenciômetros (escolheu-se uma variação de 2 unidades, o que equivale a uma variação de aproximadamente 0.7°), antes da qual ocorre leitura das configurações sem que tenha ocorrido movimento ou ocorre uma pequena variação angular antes de o veículo superar a inércia do movimento.

c) Restrição dos dados aos ângulos críticos: Uma vez obtidos os valores dos ângulos, deve-se definir seus respectivos domínios de validade para o treinamento neural. Foi necessário, então, calcular os ângulos máximos permitidos em cada articulação, também chamados de ângulos críticos. Os ângulos críticos definem a faixa de valores de cada variável que permite o controle do sistema, ou seja, para qualquer

configuração dentro dessa faixa de valores ainda é possível encontrar uma ação de controle que leve o sistema à condição desejada. Na prática, esses limites estabelecem o domínio válido das variáveis do preditor e do controlador projetados. Sabendo-se que a própria limitação física imposta ao ângulo de direção γ, de $\pm 20°$, influencia nos ângulos máximos das articulações, foi possível calcular os ângulos críticos para θ_1 e θ_2, utilizando a Equação 4.18, para tração traseira, o que resultou em $\theta_{1c} = \pm 32.0257°$ e $\theta_{2c} = \pm 54.8926°$.

Há de se ressaltar que, após a coleta, situações em que um dos ângulos de configuração ultrapassa seu respectivo valor crítico devem ser tratadas, evitando a ocorrência de *jackknife*. Uma alternativa de tratamento para tal situação pôde ser realizada via *software*, simplificando e facilitando o procedimento de coleta de dados. Assim, antes do treinamento de cada rede, os valores angulares das configurações foram restringidos aos seus respectivos ângulos críticos. Para cada trajetória coletada, foi feita uma varredura ao longo das mesmas até se achar o valor de pelo menos um dos ângulos de configuração fora de sua faixa permitida, ou seja, maior que o seu correspondente valor crítico. A partir inclusive desse valor, quaisquer configurações adicionais eram descartadas.

d) Eliminação de dados que estejam fora dos limites de treinamento desejados: A fim de facilitar o correspondente problema de controle, abordado no Capítulo 6, trechos que se tornaram côncavos, ao longo de cada trajetória, foram eliminados, sendo considerada apenas uma pequena faixa aceitável de concavidade (definiu-se uma faixa de 5° a partir da última configuração convexa, de modo a não perder uma maior quantidade de dados).

5.2.3 Preparação dos Dados para Treinamento de Redes de Diferentes Horizontes

Antes de se ter iniciado propriamente o treinamento da rede, foi fundamental promover ainda uma adequada preparação dos dados para que fosse obtida a massa de dados de cada rede com horizonte de predição característico. A tarefa de pré-processamento de dados permitiu a geração de uma base de dados bruta contendo os dados já formatados adequadamente, mas ainda agrupados por movimento. Antes do treinamento, foram extraídos dados correspondentes a cada modelo com horizonte de predição característico H, formando novas bases representativas a cada um. Durante esse processo, os padrões entrada-saída foram

extraídos da massa bruta, relacionando dados espaçados de H *ms*, de acordo com os mapeamentos e expressões abaixo:

$$\begin{aligned} &\textit{Preditor: } \left(\underline{\gamma}, \underline{\theta}, \delta\underline{\theta}\right)_k \text{ — } \left(\delta\underline{\theta} \textit{ ou } \underline{\theta}\right)_{k+h}, \\ &\textit{Controlador: } \left\{(\delta\theta_n^*)_{k+h}, \left(\underline{\theta}, \delta\underline{\theta}\right)_k\right\} \text{ — } \left(\underline{\gamma}\right)_k \end{aligned} \tag{5.3}$$

onde

$$\delta\theta_i(k) = \begin{cases} \theta_i(1), & 1 \le k \le h \\ \theta_i(k) - \theta_i(k-h), & h+1 \le k \le N \end{cases} \tag{5.4}$$

$$\delta\theta_i(k+h) = \begin{cases} \theta_i(k+h) - \theta_i(k), & 1 \le k \le N-h, \text{ dos dados extraídos da massa de dados tratada a concavidade} \\ \theta_i(k+h) - \theta_i(k), & N-h < k \le N, \text{ dos dados extraídos da massa de dados tratado o } jackknife \text{'} \end{cases}$$

N é o número de dados coletados em um movimento (ou o tamanho de cada trajetória), h = H/20 e o sobrescrito "*" indica a referência do sistema.

Além disso, foram gerados os dados simétricos aos até então obtidos e a massa de dados obtida foi unificada. Em seguida, foram incorporadas as condições singulares de giro, a partir daquelas contidas no Apêndice 1 (e das simétricas correspondentes), calculadas analiticamente através do modelo proposto nas Equações 4.11 e 4.12. Foi incorporada uma quantidade de dados de giro correspondente a aproximadamente 30% do tamanho da massa resultante do passo anterior. Na *Interface* implementada, essa porcentagem fica a critério do usuário.

A fim de evitar que algum acaso na disposição dos dados tornasse o treinamento tendencioso e de acelerar a convergência do processo de aprendizado da rede, foi feito um embaralhamento prévio dos dados. Como o Matlab® não possui nenhuma função pré-definida para embaralhamento da base de dados utilizada no treinamento, foi desenvolvido um *script* para tal fim, designado como *shuffle.m*, o qual consistia em um processo de embaralhamento por linhas de padrões entrada/saída.

A massa de dados de uma rede de horizonte H foi, então, dividida de modo a englobar certa parcela para o treinamento e outra para a validação (mais especificamente para teste como será discutido na seção 5.4.1), conforme se observa no exemplo da Figura 5.1.

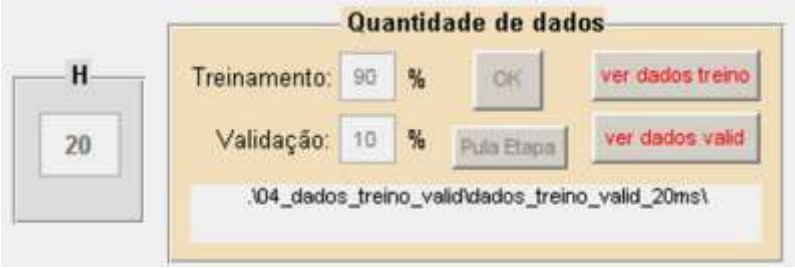


Figura 5.1 Divisão dos dados para treinamento e para teste de um preditor de horizonte de 20*ms*. As porcentagens escolhidas para treinamento e teste dependem da aplicação e do tamanho da base de dados gerada.

Em seguida, um processo de normalização dos dados de entrada foi posto em prática. Assim, os dados normalizados foram mapeados para dentro do domínio válido da função de ativação utilizada, visando, sempre que possível, evitar sua região de saturação, o que saturaria as respostas dos neurônios. Os desempenhos de duas técnicas de normalização foram comparados. Uma delas era fornecida e implementada pelo próprio toolbox *nnet* do Matlab® e consistia no seguinte mapeamento $[x_{min}, x_{max}] \rightarrow [y_{min}, y_{max}]$, baseado nos valores máximo, x_{max}, e mínino, x_{min}, de cada linha de uma matriz de entrada *x*, conforme a Equação 5.5. A outra técnica será explicada na sequência em maiores detalhes.

$$y = (y_{max} - y_{min}) * \frac{(x - x_{min})}{(x_{max} - x_{min})} + y_{min} \tag{5.5}$$

As redes neurais são sensíveis à escala das variáveis de entrada. Assim, se a magnitude de alguns valores de entrada diferir muito da de outros, a rede pode erroneamente atribuir uma maior importância a valores maiores. Desse modo, com o intuito de se eliminar, da massa de dados, ocorrências espúrias de valores que, com a normalização habitual, comprimiriam os valores dos demais dados, tirando suas relevâncias numéricas, uma nova técnica de normalização foi proposta durante esse trabalho. O algoritmo é mostrado abaixo:

Algoritmo de Normalização Proposto

1- Calcular a média M_1 de todas as T amostras (elementos do vetor de entrada);

2- Definir as variáveis: $porcent$ e M_{limiar}.

Loop (i = 1, ...): $\begin{cases} Critério\ de\ parada\ 1\text{: } quantidade\ (\mathrm{T}_i) \leq porcent\ \%\ (\mathrm{T}) \\ Critério\ de\ parada\ 2\text{: } M_{atual} \leq M_{limiar} \end{cases}$.

a) Identificar os elementos amostrais, T_i, maiores que o somatório das médias calculadas $M_i, \ldots, M_1$;

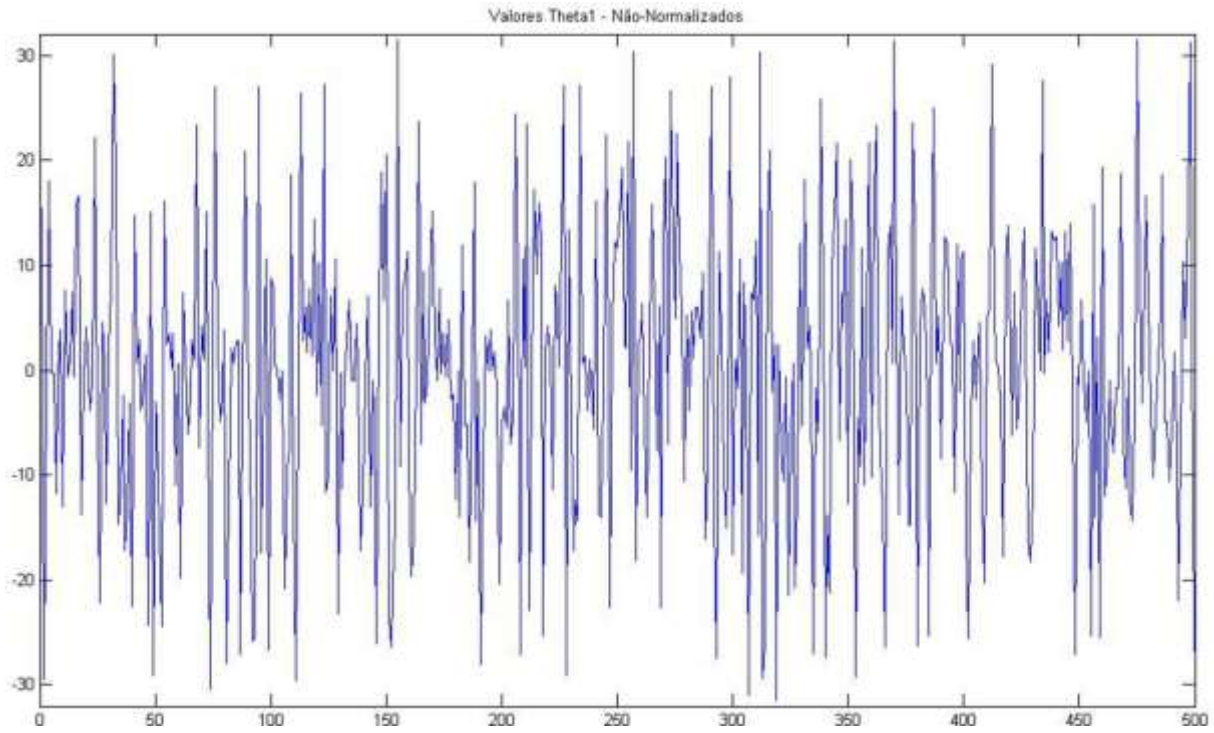
b) Calcular a média M_{i+1} da diferença entre os T_i elementos e o somatório das

médias calculadas $M_i, \ldots, M_1$, porém levando em consideração a quantidade total T de amostras.

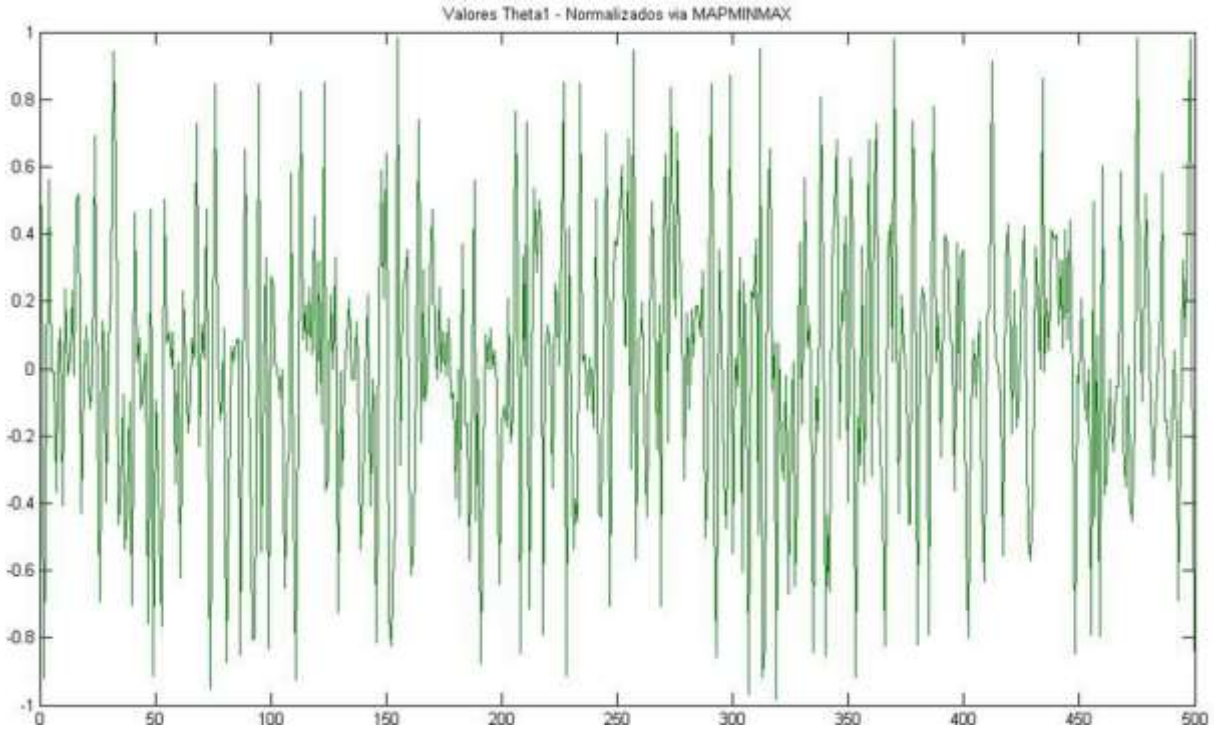
Fim Loop;

3- Saturar vetor com base no seu maior elemento, excluído os elementos presentes em T_{atual};
4- Normalizar vetor usando como valor de normalização: $m = \sum_i M_i$, mapeamento $[x \rightarrow y] : [(x_{min}, x_{max}) \rightarrow (-1, +1)]$.

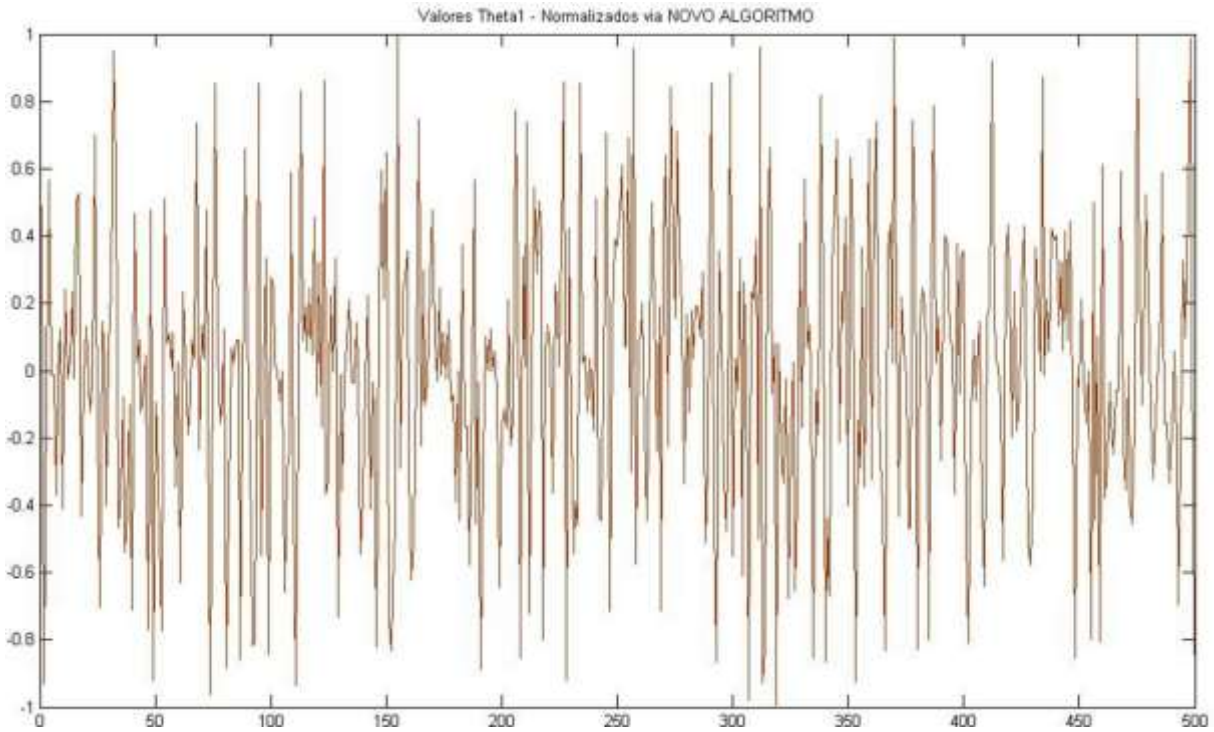
A Figura 5.2 mostra as distribuições amostrais dos ângulos de configuração originais, extraídos da base para treinamento de um preditor de horizonte de 40*ms*, e dos ângulos após a aplicação dos algoritmos de normalização discutidos. O eixo vertical indica o valor do ângulo em graus, enquanto que o horizontal, o índice do valor.



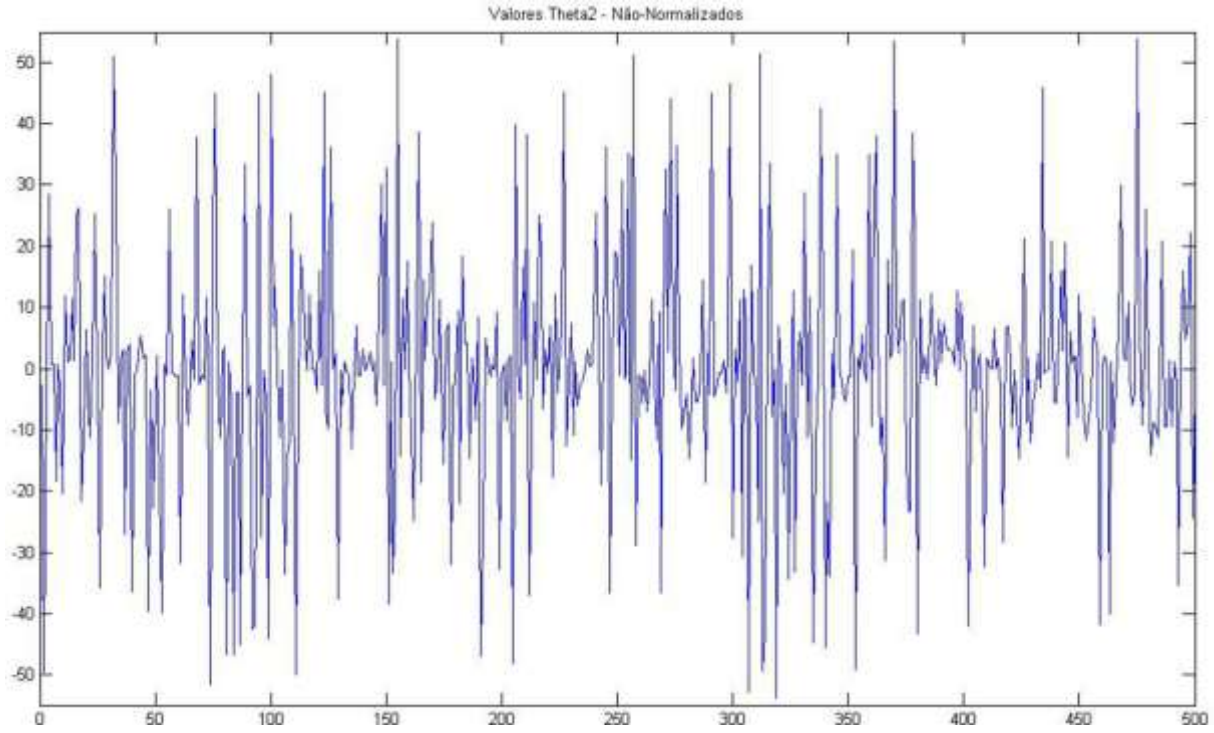
(a)


Valores Theta1 - Normalizados via MAPMINMAX
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
50
100
150
200
250
300
350
400
450
500

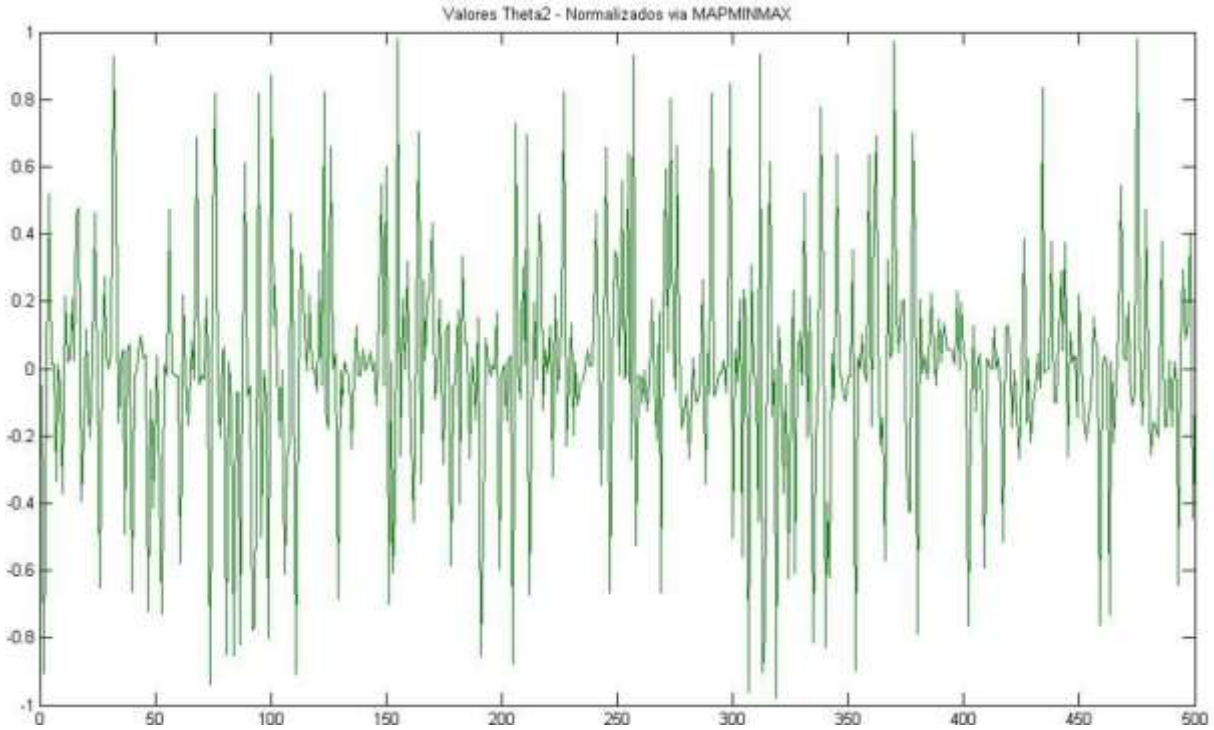
(b)


Valores Theta1 - Normalizados via NOVO ALGORITMO
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
50
100
150
200
250
300
350
400
450
500

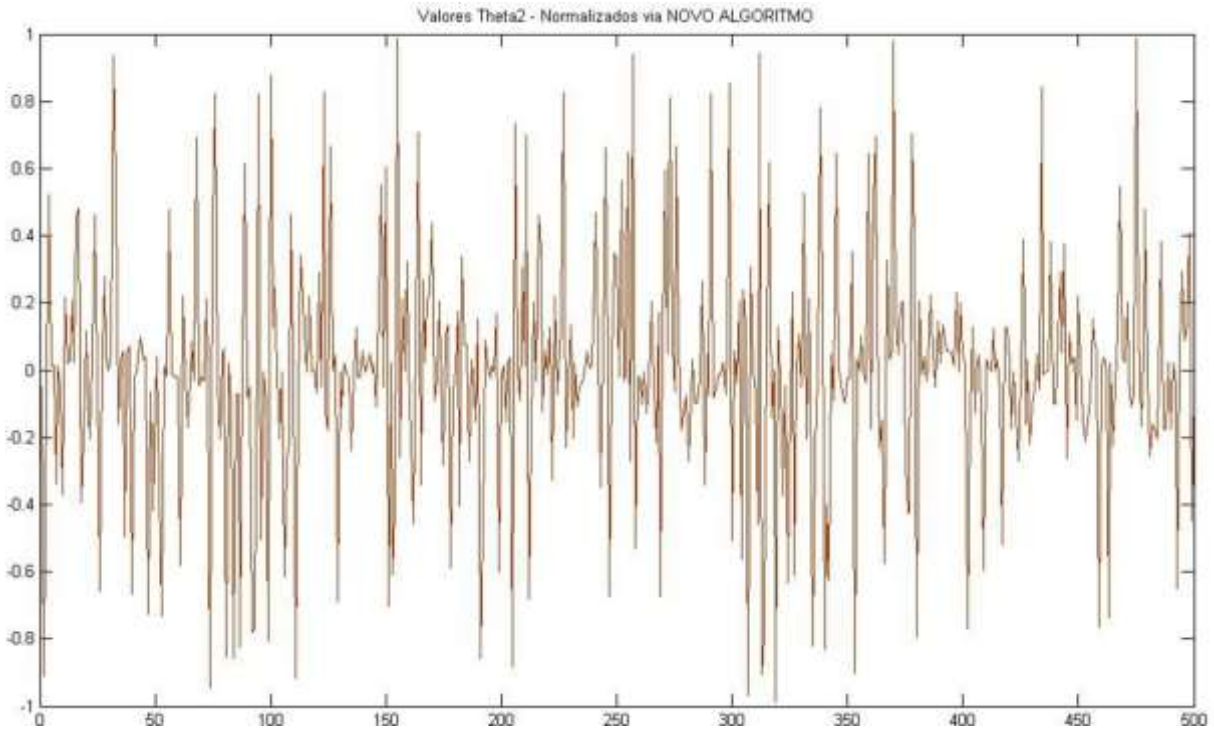
(c)


Valores Theta2 - Não-Normalizados
50
40
30
20
10
0
-10
-20
-30
-40
-50
0
50
100
150
200
250
300
350
400
450
500

(d)


Valores Theta2 - Normalizados via MAPMINMAX
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
50
100
150
200
250
300
350
400
450
500

(e)



(f)

Figura 5.2 Distribuições amostrais dos ângulos de configuração, antes e após as normalizações, para um preditor de horizonte de 40*ms*. (a) 500 primeiros valores de θ_1 antes das normalizações. (b) 500 primeiros valores de θ_1 após normalização usando a Equação 5.5. (c) 500 primeiros valores de θ_1 após normalização usando o algoritmo proposto. (d) 500 primeiros valores de θ_2 antes das normalizações. (e) 500 primeiros valores de θ_2 após normalização usando a Equação 5.5. (f) 500 primeiros valores de θ_2 após normalização usando o algoritmo proposto.

Percebe-se que as distribuições amostrais dos ângulos de configuração após as normalizações, para cada preditor, praticamente não se alteraram em comparação às originais correspondentes, mantendo a importância numérica dos valores originais, o que comprova o bom desempenho do algoritmo proposto e a correta escolha da função de normalização utilizada.

A Figura 5.3 mostra a parte da *Interface* destinada à etapa de pré-processamento e preparação dos dados. Essa ferramenta desenvolvida, em geral, torna bastante elementar a averiguação das técnicas utilizadas ao longo deste trabalho e permite, de maneira mais simples, a reprodução dos resultados obtidos em cada etapa do projeto de modelagem e controle neurais.

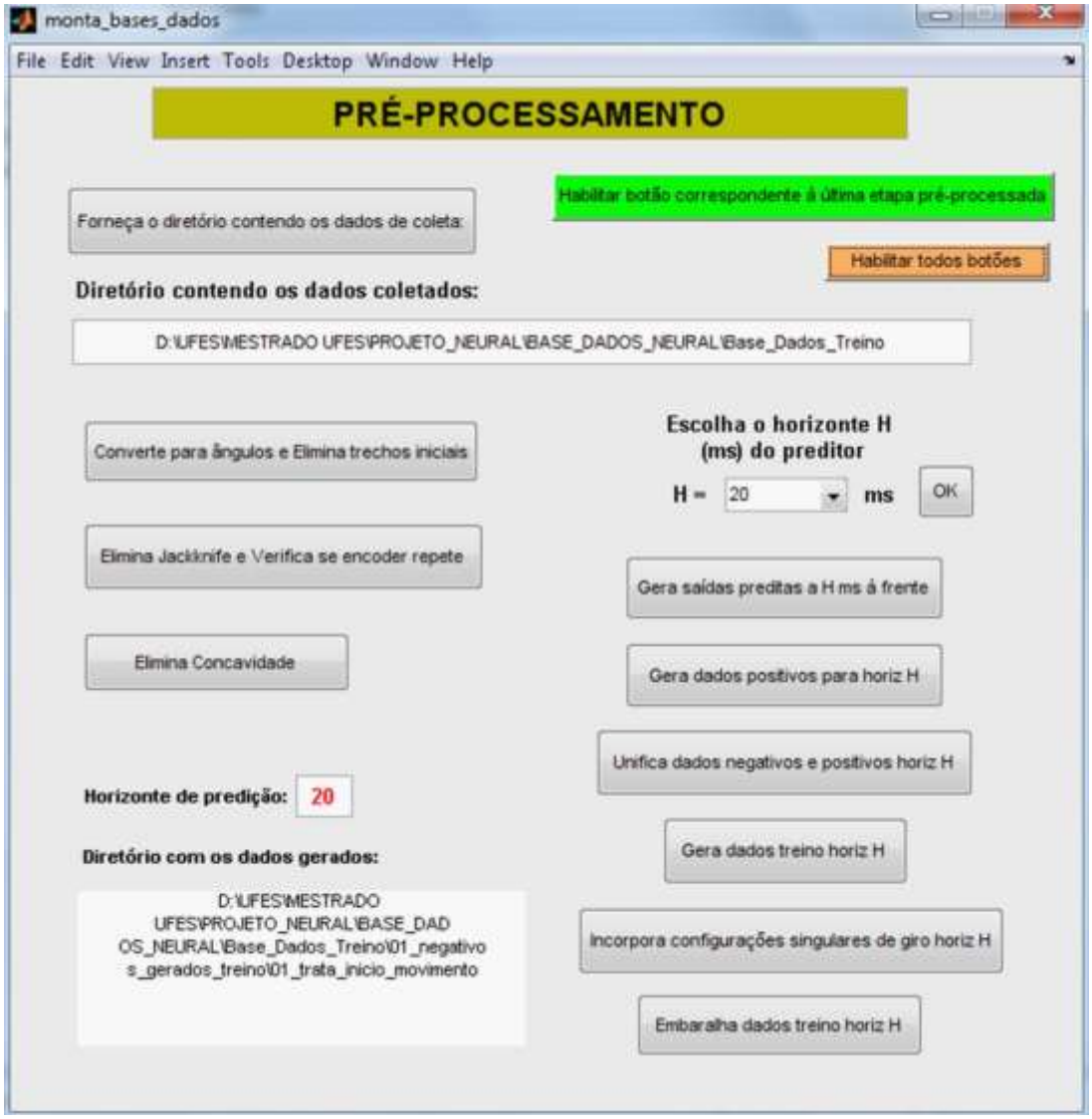


Figura 5.3 *Interface* destinada à etapa de pré-processamento e preparação dos dados (*monta_bases_dados.m, .fig*).

Do ponto de vista da constituição das bases de dados para treinamento de cada rede (seja preditor ou controlador), o processo aparentemente complexo e trabalhoso revelou-se deveras simplificado com o emprego da estratégia de particionar e combinar os domínios dos ângulos de configuração e de direção, de modo a gerar condições iniciais de movimento ao RMMA. Os procedimentos aplicados às bases de dados geradas possibilitaram a formação de um conjunto de informações para o treinamento de cada preditor aparentemente suficiente para tal objetivo. Esse mesmo fato será averiguado no Capítulo 6 durante o projeto do controlador.

5.2.4 Escolha da Topologia e Parametrização de Preditores

Uma vez que a *Interface* criada possibilita a determinação da melhor topologia e parametrização de rede, os desempenhos de diversas estruturas (topologias + parametrizações) candidatas, para um mesmo horizonte de predição, foram comparados, sendo possível, assim, estabelecer uma a ser testada para diferentes horizontes. A comparação entre os desempenhos relativos das diversas redes foi feita não só durante seus treinamentos, mas também quando a elas aplicados os procedimentos de validação a serem explicados na Seção 5.4. A Tabela 5.1 mostra as características de cada topologia de rede comparada. A Tabela 5.2 mostra os resultados obtidos pela estratégia[23] adotada para se avaliar o desempenho de cada estrutura neural.

Tabela 5.1 Estruturas de preditores para um mesmo horizonte de predição (H = 40*ms*).

	Estrutura 1	**Estrutura 2**	**Estrutura 3**	**Estrutura 4**
Número de camadas ocultas	*3*	*2*	*2*	*2*
Número de neurônios por camada oculta	*7, 11 e 13*	*7 e 11*	*7 e 11*	*7 e 11*
Função de treinamento	*trainscg*	*trainscg*	*traingdx*	*trainscg*
Número de épocas estabelecido	*1000*	*1000*	*3000*	*1000*
Erro médio quadrático (*mse*) alcançado	*0.036°*	*0.035°*	*0.0406°*	*0.0409°*
Domínio e Quantidade de dados de giro (% dados treino)	*[-54°:0.1:-1°] (30%)*	*[-54°:0.1:-1°] (30%)*	*[-54°:0.1:-1°] (30%)*	*[-54.8° : 0.351562 : -0.351562°] (20%)*

Parâmetros em comum: Função de ativação por camada oculta: sigmoidal bipolar – *tansig* / Função de ativação na camada de saída: linear – *purelin* / Função de aprendizado: *learngdm*. / Erro médio quadrático objetivado (*Goal*) nulo.

Tabela 5.2 Desempenho de cada estrutura de preditor no processo de validação (aproximação de trajetórias a partir de uma condição angular inicial).

Estruturas neurais	**Quantidade de trajetórias onde se obteve menor erro (*rmse*) para θ_1**	**Quantidade de trajetórias onde se obteve menor erro (*rmse*) para θ_2**	**Média dos menores erros (*rmse's*) por trajetória para θ_1**	**Média dos menores erros (*rmse's*) por trajetória para θ_2**
Estr. 1	*153*	*205*	*2.2511*	*1.5052*
Estr. 2	*257*	*196*	*2.3317*	*2.3984*
Estr. 3	***359***	***381***	*1.7775*	*1.6454*
Estr. 4	*185*	*172*	*2.5103*	*1.6318*

[23] Detalhes complementares dessa abordagem serão explicados na Tabela 6.5.

Vale ressaltar que o procedimento utilizado para a escolha da melhor estrutura de preditores não será explicado ainda nessa seção. Far-se-á a abstração da referida estratégia ao se considerar somente, por enquanto, os dados evidenciados na Tabela 5.2. Essa tabela se assemelha muito à Tabela 5.5, já que o processo empregado de validação e a estratégia destinada a medir o desempenho de cada rede neural são os mesmos em ambos os casos, porém, no caso a ser mostrado posteriormente na Seção 5.4.3, trabalha-se com a estrutura definida nessa seção aplicada à preditores com horizontes de predição, H, diferentes.

Uma mesma estrutura de rede foi criada para diferentes horizontes de predição e o desempenho dos respectivos preditores foi analisado durante seus treinamentos (Seção 5.3) e suas validações (Seção 5.4). As características da melhor estrutura a ser usada para cada preditor são mostradas abaixo:

Estrutura 3:

- Domínio e Quantidade de dados de giro (% dados treino): [-54° : 0.1° : -1°] e (30%);
- Número de camadas ocultas: 2;
- Número de neurônios por camada oculta: 7 e 11, respectivamente;
- Função de criação da rede multicamada *feedforward*: *newff*;
- Função de ativação por camada oculta: sigmoidal bipolar – *tansig*;
- Função de ativação na camada de saída: linear – *purelin*;
- Função de treinamento: *traingdx,* o qual se baseia no gradiente descendente com uma taxa de aprendizado adaptativa e adição de um componente de momento, ambos de modo a evitar instabilidade e garantir convergência, conforme explicados no Apêndice 1;
- taxa de aprendizado inicial (*lr*) = 0.05 / constante de momento (*mc*) = 0.9;
- Função de aprendizado: *learngdm*, aprendizado com adição do componente momento;
- Funções de pré-processamento: *fixunknowns*, *mapminmax*, *removeconstantrows*;
- Funções de pós-processamento: *removeconstantrows*, *mapminmax*;
- Função de *performance*: *mse*;
- Sem Parada Antecipada (*Early Stopping*)[24]: Esse procedimento não foi adotado, pois o treinamento de cada rede estava sendo paralisado com um número relativamente pequeno de épocas, o que acarretava no não-aprendizado da rede.
- Função de inicialização aleatória dos pesos e *biases* da rede.

[24] Procedimento onde o processo de aprendizado de uma rede é constantemente checado pela aplicação de um subconjunto de validação, extraído da própria massa de treinamento, sendo finalizado quando começar a ocorrer elevação do erro quadrático (frente ao subconjunto de validação) após certo número sucessivo de épocas.

5.3 Treinamento dos Preditores

Grande parte deste trabalho foi desenvolvida fazendo uso do *software* Matlab®, seu ambiente de criação de interfaces gráficas, o *Guide*, e seu *toolbox* de redes neurais, o *nnet*. De modo a aproveitar as versatilidades da *Interface* implementada e do *nnet*, foram criados artifícios para estruturar o treinamento tornando-o um processo mais transparente e de mais fácil entendimento e, portanto, permitindo ao usuário concentrar esforços no entendimento da metodologia de projeto de preditores e controladores e não somente na programação de algoritmos de treinamento. O algoritmo de aprendizado supervisionado neural utilizado foi o *Backpropagation* e suas variações, discutido no Apêndice 1. Além disso, foi usada a *Interface* para gerar os resultados de cada procedimento aplicado à massa de dados, aos preditores e aos controladores.

Na presente etapa, foi utilizado o modo *batch* de treinamento. Apesar de requerer um maior espaço de armazenamento de dados, a capacidade do computador onde os experimentos foram realizados se revelou mais do que suficiente. Com esse tipo de treinamento, pôde-se ter uma melhor estimativa do vetor gradiente do erro, o que contribuiu para um treinamento mais estável.

Para que o treinamento convergisse adequadamente dentro do número de épocas estabelecido, verificou-se que, à medida que se aumentava o horizonte do preditor, era necessário ou estabelecer um maior número de épocas, mantendo aproximadamente o mesmo erro, ou estabelecer um maior erro desejado, sem alterar significativamente o número de épocas. Porém, no primeiro caso, o aumento do número de épocas poderia gerar o *overfitting* (memorização) da rede, além do fato de não assegurar a convergência da rede para um erro pequeno estabelecido. Já no segundo caso, um maior erro desejado poderia ser atingido em um número pequeno de épocas, dependendo do horizonte de predição utilizado, o que acarretaria no não-aprendizado da rede (*underfitting*).

Uma estratégia mais viável seria estabelecer inicialmente um número de épocas que não gerasse o *overfitting* da rede e um erro idealmente nulo, observando o erro, mínimo, realmente atingido dentro desse intervalo. Caso o erro atingido fosse ainda grande, a especificação do número de épocas poderia ser relaxada, porém sempre de modo a evitar o *overfitting* e a garantir o aprendizado da rede. O desempenho de cada preditor seria avaliado com base em um critério fixo (nesse caso, o número de épocas), observando-se a relação entre o erro atingido por cada preditor e suas respectivas precisões.

Em suma, o número adequado de épocas, em cada caso, foi, então, estimado experimentalmente, após vários e sucessivos treinos com a rede, de modo a evitar um valor, o qual a rede relativamente considerasse alto, ocasionando sua especialização no conjunto de dados de entrada, assim como um valor, o qual fosse considerado baixo e não o suficiente para garantir o aprendizado[25]; com isso, foi possível efetuar o treinamento considerando o menor erro entre as saídas obtidas e esperadas da rede dentro do limite de épocas estabelecido.

Para cada preditor de horizonte diferente, o erro médio quadrático atingido durante o seu treinamento foi bem menor que a precisão (menor variação angular não-nula, considerando o horizonte de predição) em que os dados de sua base foram gerados, tendo em vista que cada rede convergia muito rapidamente ao se estipular um erro de treinamento equivalente a essa precisão, o que ocasionava no não-aprendizado da rede. Esse fato é ilustrado na Tabela 5.3. Como exemplo, para H de 40*ms*, a precisão era de aproximadamente 0.7° e o erro alcançado foi de 0.0406°.

Tabela 5.3 Número de épocas e erro médio quadrático para alguns preditores durante seus treinamentos.

Horizonte H (*ms*)	**N° Máx Épocas**	**N° Épocas Atingido**	**Goal (*mse*)**	**Erro (*mse*)**	**Precisão**	**Tamanho Base Dados**
40	*3000*	*3000*	*0°*	*0.0406°*	*0.7°*	*81232*
80	*3000*	*3000*	*0°*	*0.0664°*	*1.4°*	*80852*
120	*3000*	*3000*	*0°*	*0.0972°*	*2.1°*	*80448*

A Figura 5.4 ilustra um aplicativo gráfico para treinamento das redes neurais, provida pelo *nnet*, e o esquemático de treinamento de um preditor de horizonte de predição H. Já a Figura 5.5 mostra a parte da *Interface* implementada destinada a criar, parametrizar, treinar e testar a generalidade de um preditor de horizonte escolhido, H.

[25] A comprovação experimental da estimação de um número adequado de épocas foi feita após a análise dos resultados obtidos quando aplicados os procedimentos de validação neural detalhados no decorrer deste capítulo.

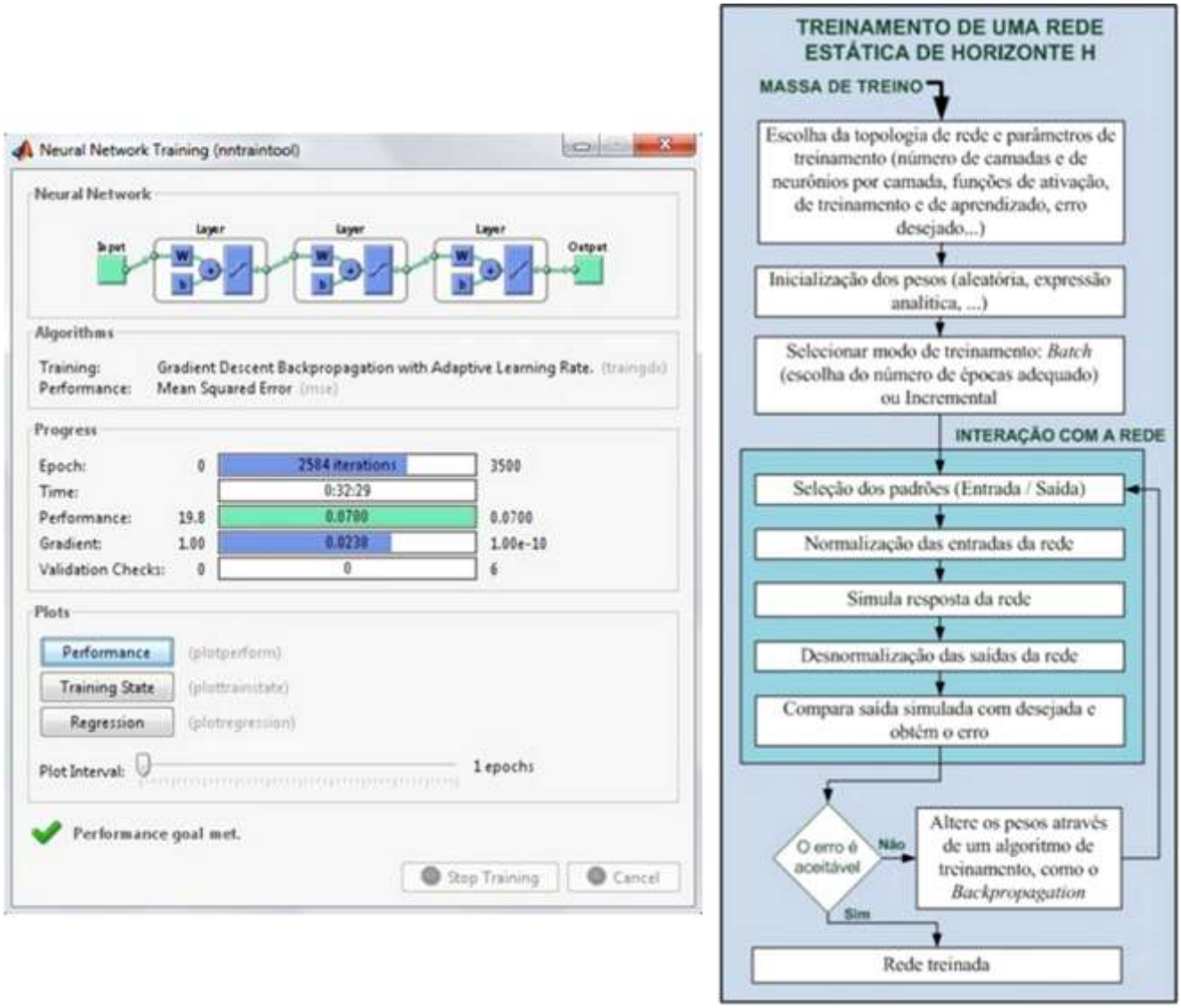


Figura 5.4 Ilustração do aplicativo gráfico para treinamento (à esquerda) e do esquemático de treinamento[26] (à direita) de um preditor de horizonte H.

Em relação à estruturação da rede, a topologia empregada para constituição de cada preditor e a parametrização aplicada para o treinamento se demonstraram adequados, em virtude dos resultados obtidos dentro do esperado, suficientes a consolidar o aprendizado neural.

Na fase de treinamento, foram observados os diversos comportamentos de cada preditor em relação a alguns algoritmos de treinamento (apresentados na Tabela 5.4), derivados do *Backpropagation*, já implementados pelo *toolbox nnet* do Matlab® e disponíveis na *Interface* criada, a fim de se estudar a melhor possibilidade a ser aplicada ao aprendizado

[26] Nesse caso, a variável de critério de parada utilizada foi o erro (*mse*).

da rede, segundo um número de épocas apropriado estabelecido e erro entre as saídas aceitável. A função de treinamento escolhida (*traingdx*), conforme mencionada no decorrer deste capítulo, se revelou relativamente a mais apta a tal objetivo e a convergência foi garantida dentro dos limites impostos.

Tabela 5.4 Alguns algoritmos de treinamento.

Função	***Técnica***
traingdx	*Variable Learning Rate Backpropagation*
trainscg	*Scaled Conjugate Gradient Backpropagation*
trainrp	*Resilient Backpropagation (Rprop)*
trainlm	*Levenberg-Marquardt Backpropagation*

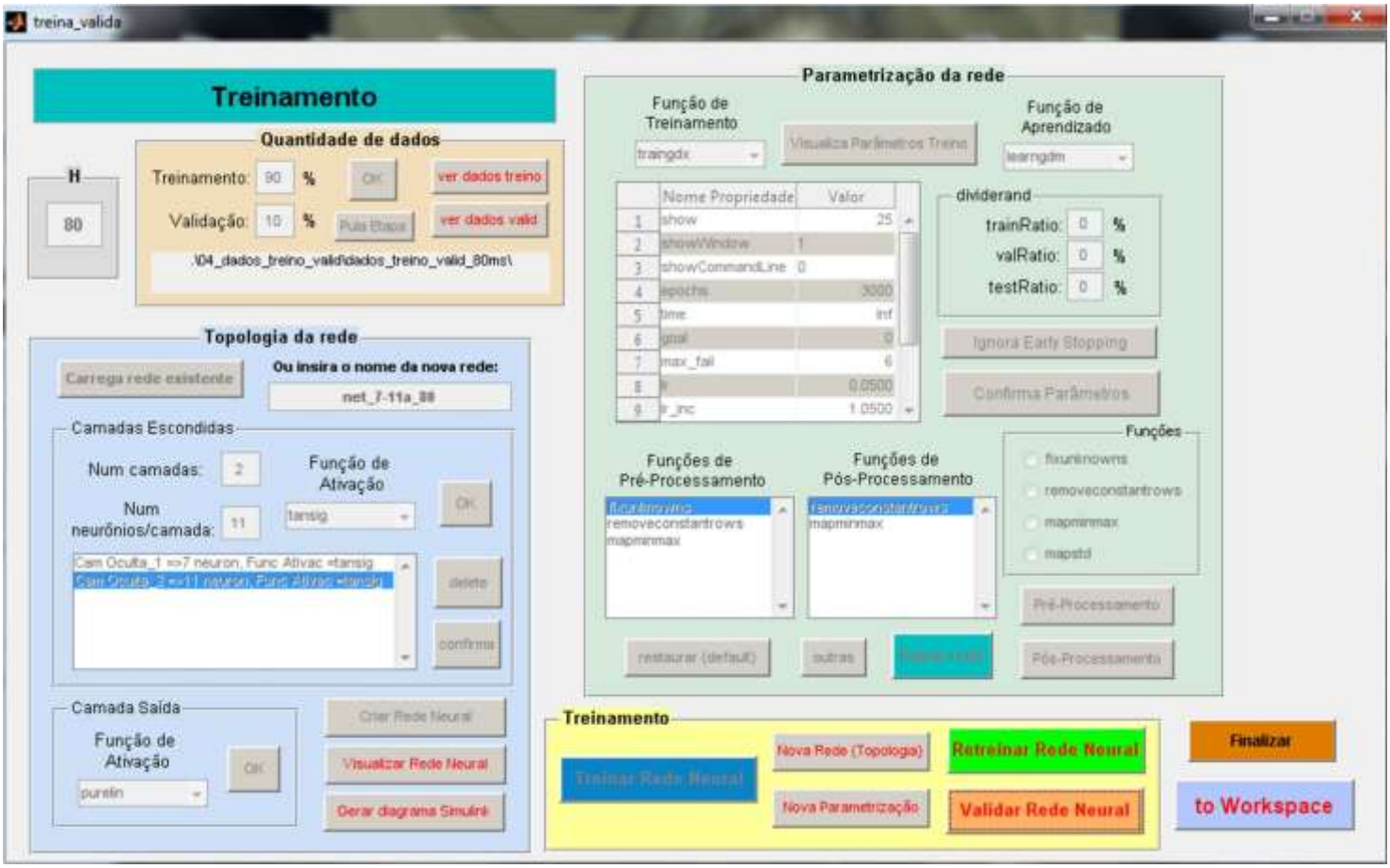


Figura 5.5 Ilustração da parte da *Interface* implementada destinada a criar, a parametrizar, a treinar e a testar a generalidade de um preditor de horizonte H (*treina_valida.m, .fig*).

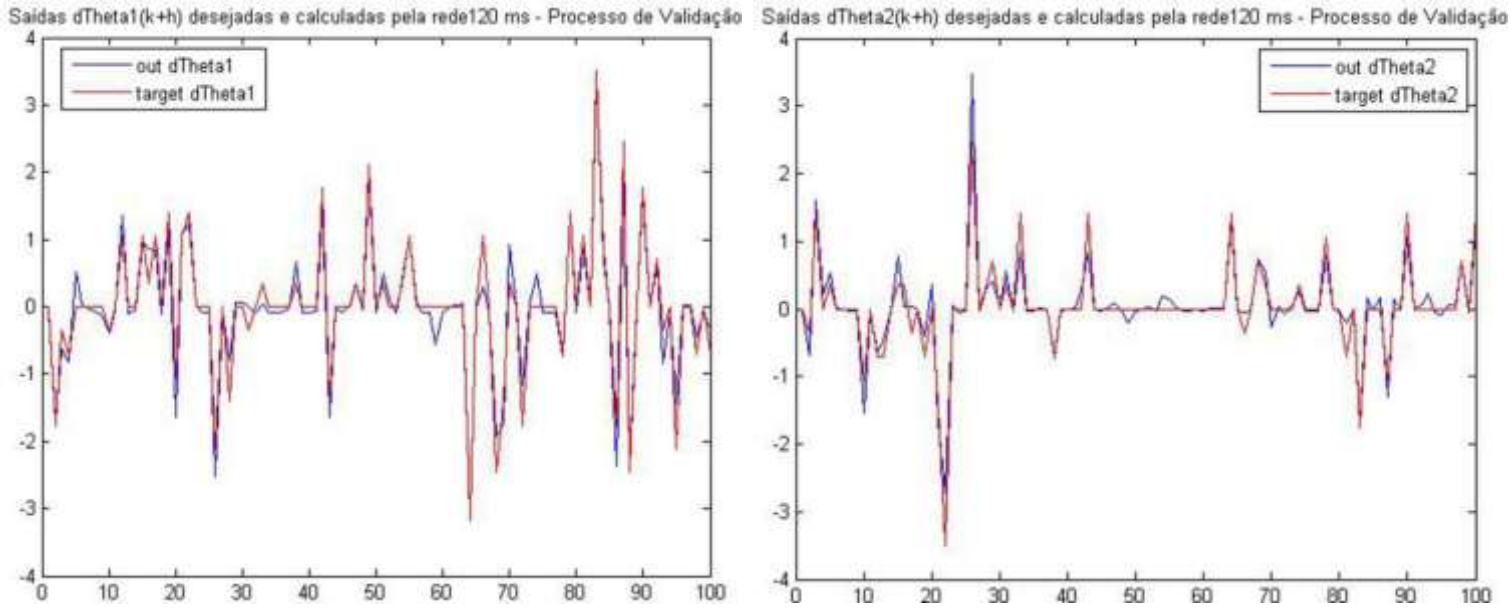
5.4 Validação dos Preditores

Após o treinamento de um preditor com certo horizonte de predição, um conjunto de dados não utilizado previamente no seu treinamento foi submetido à rede a fim de avaliar, a cada novo padrão, a sua capacidade de generalização. Esse conjunto era composto ou pela parcela da base de dados contendo os padrões de entrada-saída embaralhados destinados à validação (aqueles 10%, nesse caso) – vide seção 5.4.1 – ou por dados oriundos de novas trajetórias coletadas (diferentes daquelas usadas para compor a massa de treinamento do preditor) – vide seção 5.4.2.

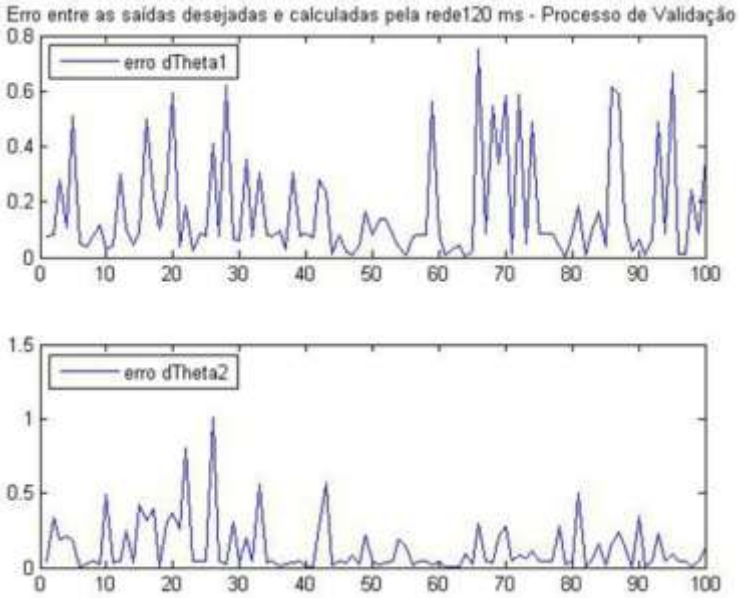
5.4.1 Teste de Generalidade

Na primeira abordagem, verifica-se o quão bem os preditores treinados respondem individualmente a cada novo padrão entrada-saída extraído da massa embaralhada, não correspondente a uma específica trajetória. Esse processo, denominado Teste de Generalidade, permite ao usuário balizar, iterativamente, escolhas iniciais de topologias, parâmetros e horizontes de predição na busca de uma solução satisfatória nos problemas de modelagem e controle. Durante essa etapa, é verificado, para todos os preditores, se os erros médios quadráticos (*mse*), entre a saída calculada e a objetivada, são muito próximos aos correspondentes erros alcançados em seus treinamentos.

Como exemplo desse processo, para um preditor com horizonte H de 120*ms*, o erro médio quadrático obtido (0.086°) foi aproximadamente igual ao atingido em seu treinamento (0.097°). O mesmo fato fora observado para os demais preditores com horizontes diferentes. Na Figura 5.6, são apresentados os erros e as comparações entre cada saída obtida por um preditor treinado, de horizonte de 120*ms*, e a saída desejada correspondente, durante o seu Teste de Generalidade. Nos gráficos, os resultados em graus (eixo vertical) são mostrados para cada padrão (eixo horizontal).



(a)



(b)

Figura 5.6 Resultados do Teste de Generalidade para o preditor de horizonte de 120*ms*. (a) Comparação entre a saída, $\delta\theta_1(k+h)$, calculada e desejada (à esquerda). Comparação entre a saída, $\delta\theta_2(k+h)$, calculada e desejada (à direita). (b) Erro absoluto entre os valores obtidos e objetivados, para cada saída – exibição dos 100 primeiros valores, em cada caso.

5.4.2 Validação Ponto-a-Ponto

Na segunda abordagem, o objetivo consistia em avaliar se cada preditor seguia trajetórias convexas descritas pelo RMMA. Verificou-se o quão bem os preditores

reproduziram o comportamento de uma trajetória, respondendo individualmente a cada padrão entrada-saída dessa trajetória. O esquema da Figura 5.7 ilustra os processos discutidos.

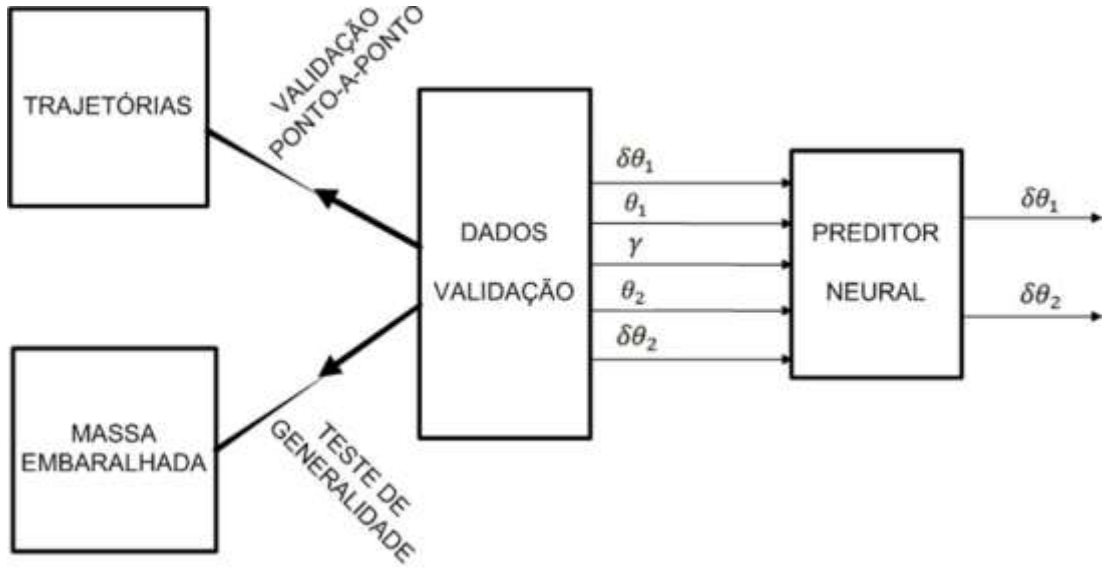
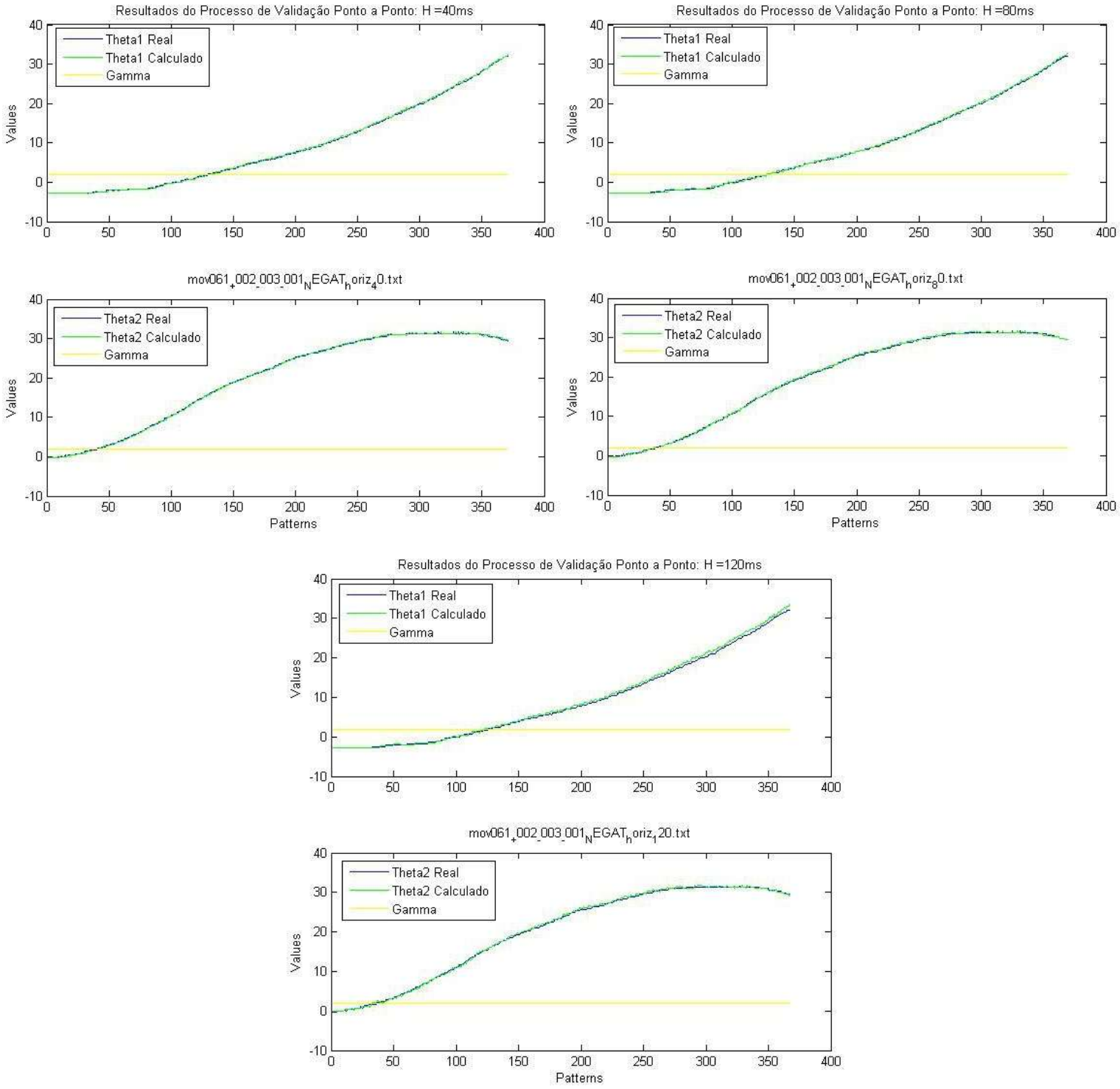


Figura 5.7 Esquema do Teste de Generalidade ou da Validação Ponto-a-Ponto de um preditor neural de horizonte H qualquer.

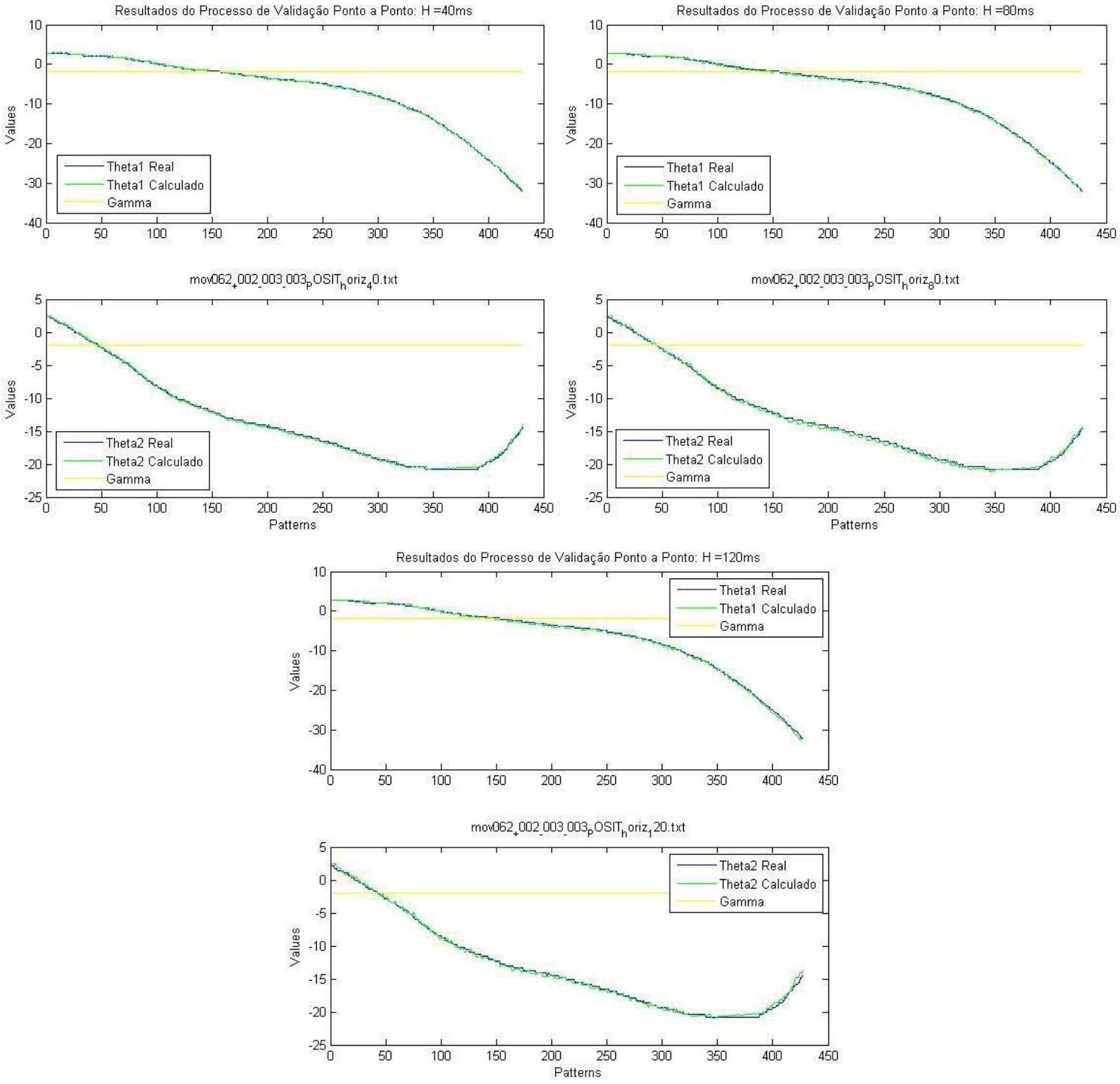
A Figura 5.8 mostra os resultados do processo de Validação Ponto-a-Ponto para algumas trajetórias escolhidas. Nos gráficos, as retas em amarelo correspondem aos valores do ângulo de direção do RMMA, mantidos, nesse caso, constantes; os valores em azul representam os diversos valores dos ângulos de configuração correspondentes às trajetórias reais; e os valores em verde representam a resposta de um preditor quando a ele aplicada cada condição $(\gamma, \theta_1, \theta_2, \delta\theta_1, \delta\theta_2)$ proveniente de trajetórias reais. Os resultados em graus (eixo vertical - *Values*) são mostrados para cada padrão (eixo horizontal - *Patterns*). O tempo total decorrido em cada trajetória é calculado multiplicando-se o número de iterações (equivalente ao número de padrões) pelo tempo de amostragem (nesse caso igual a $20ms$).

Trajetória 1: condição inicial ($\gamma = +2°, \theta_1 = -3°, \theta_2 = -1°$):



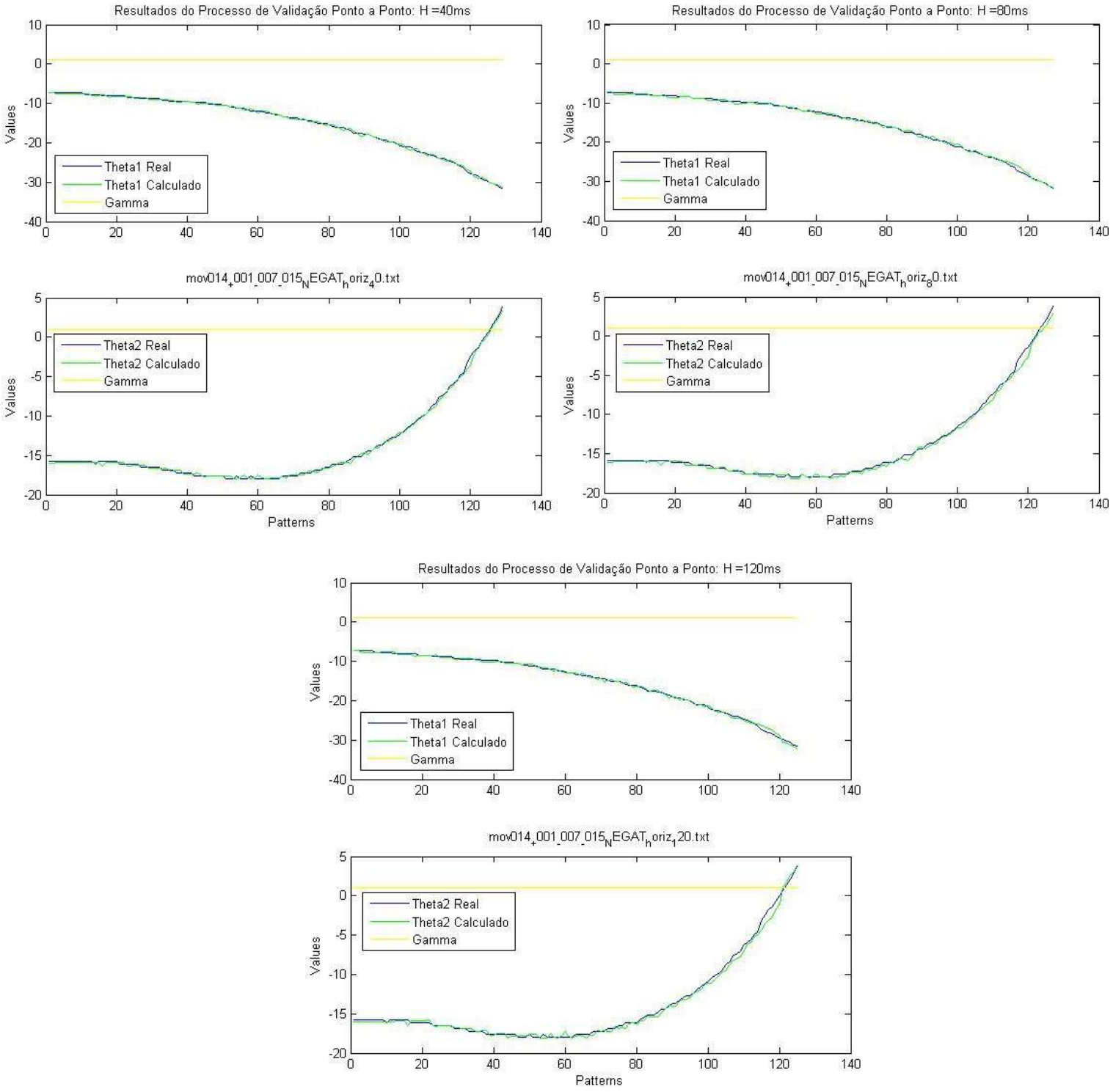
(a)

Trajetória 2: condição inicial ($\gamma = -2°, \theta_1 = +3°, \theta_2 = +3°$):



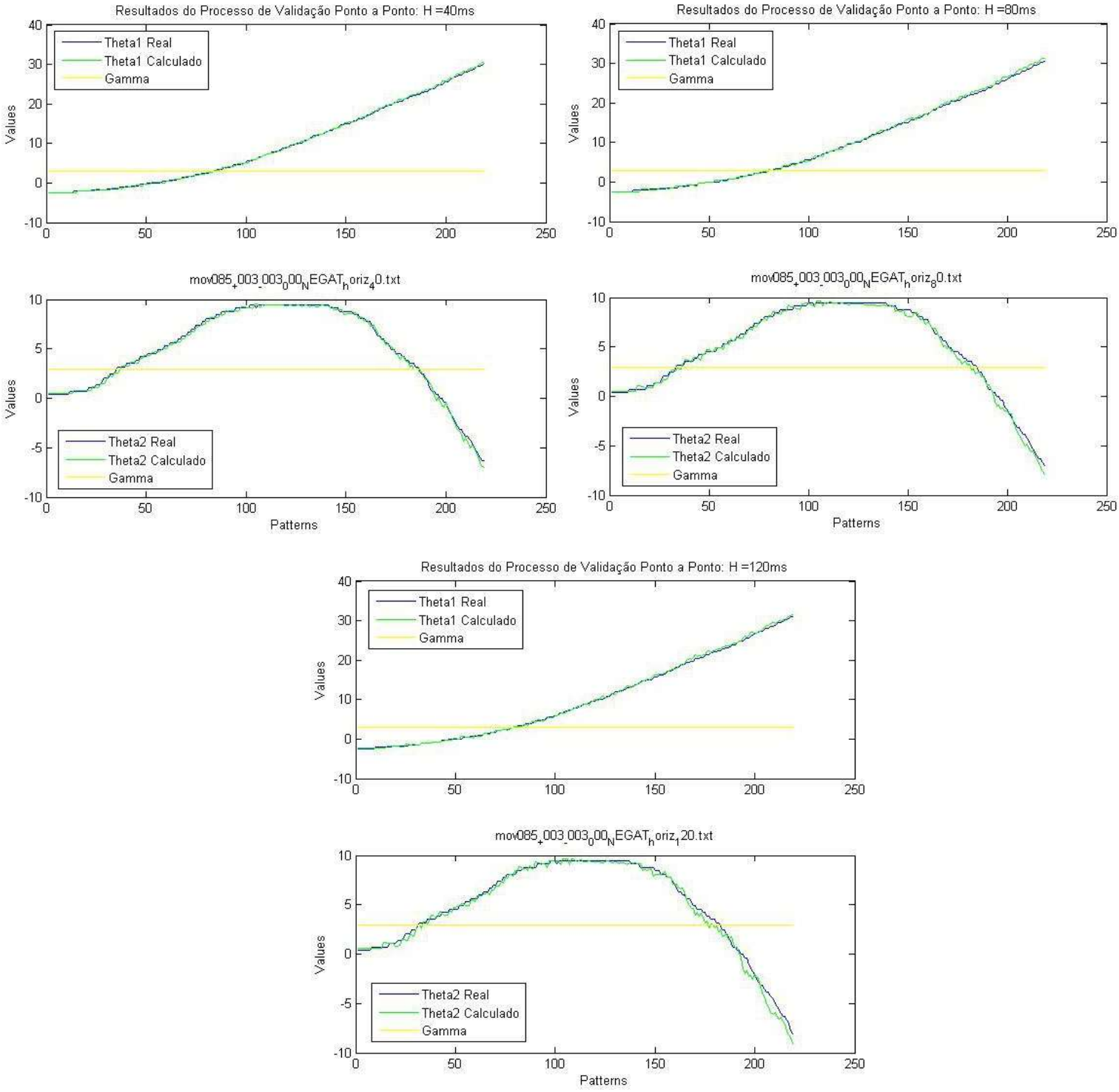
(b)

Trajetória 3: condição inicial ($\gamma = +1°, \theta_1 = -7°, \theta_2 = -15°$):



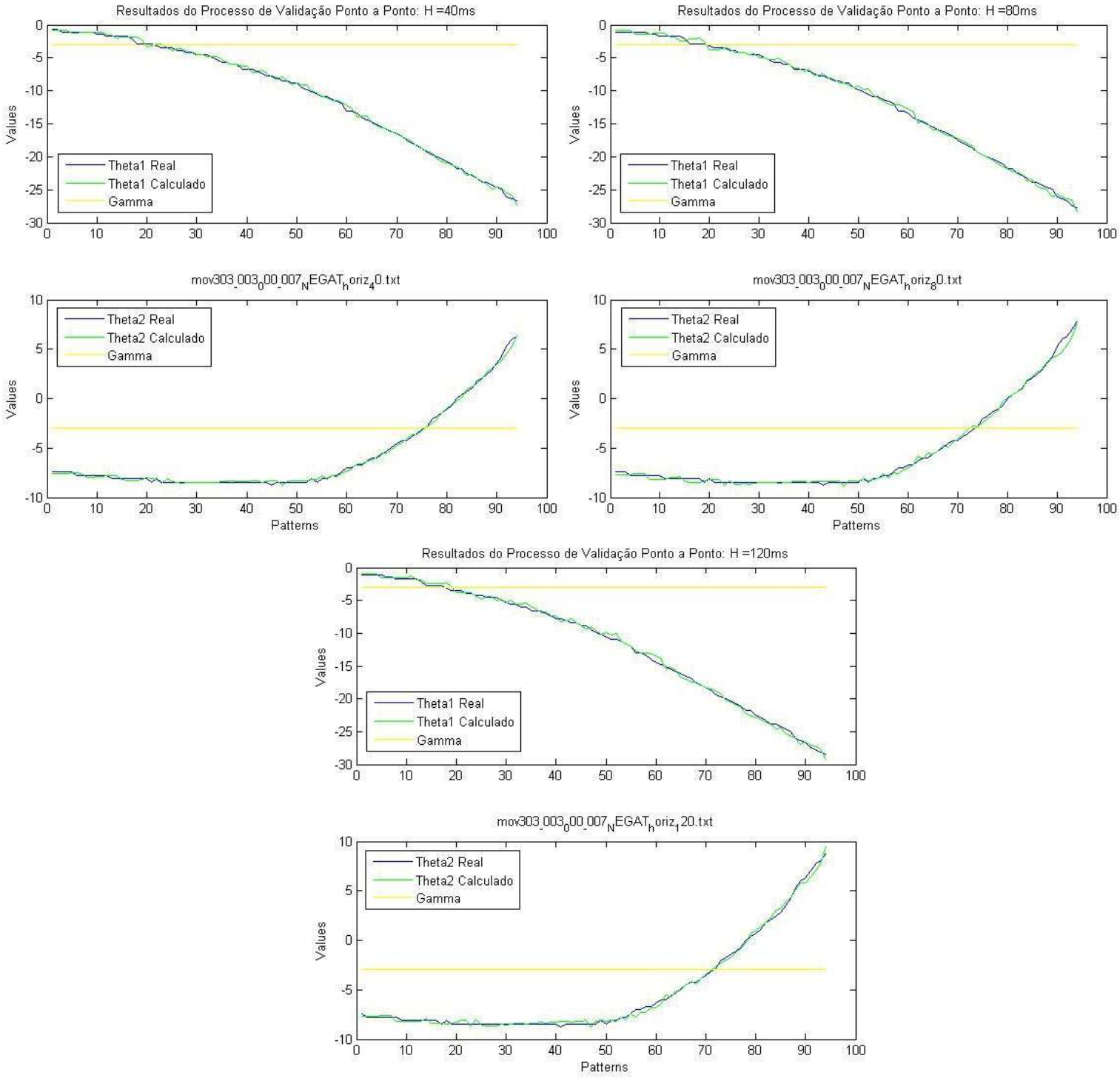
(c)

Trajetória 4: condição inicial ($\gamma = +3°, \theta_1 = -3°, \theta_2 = 0°$):



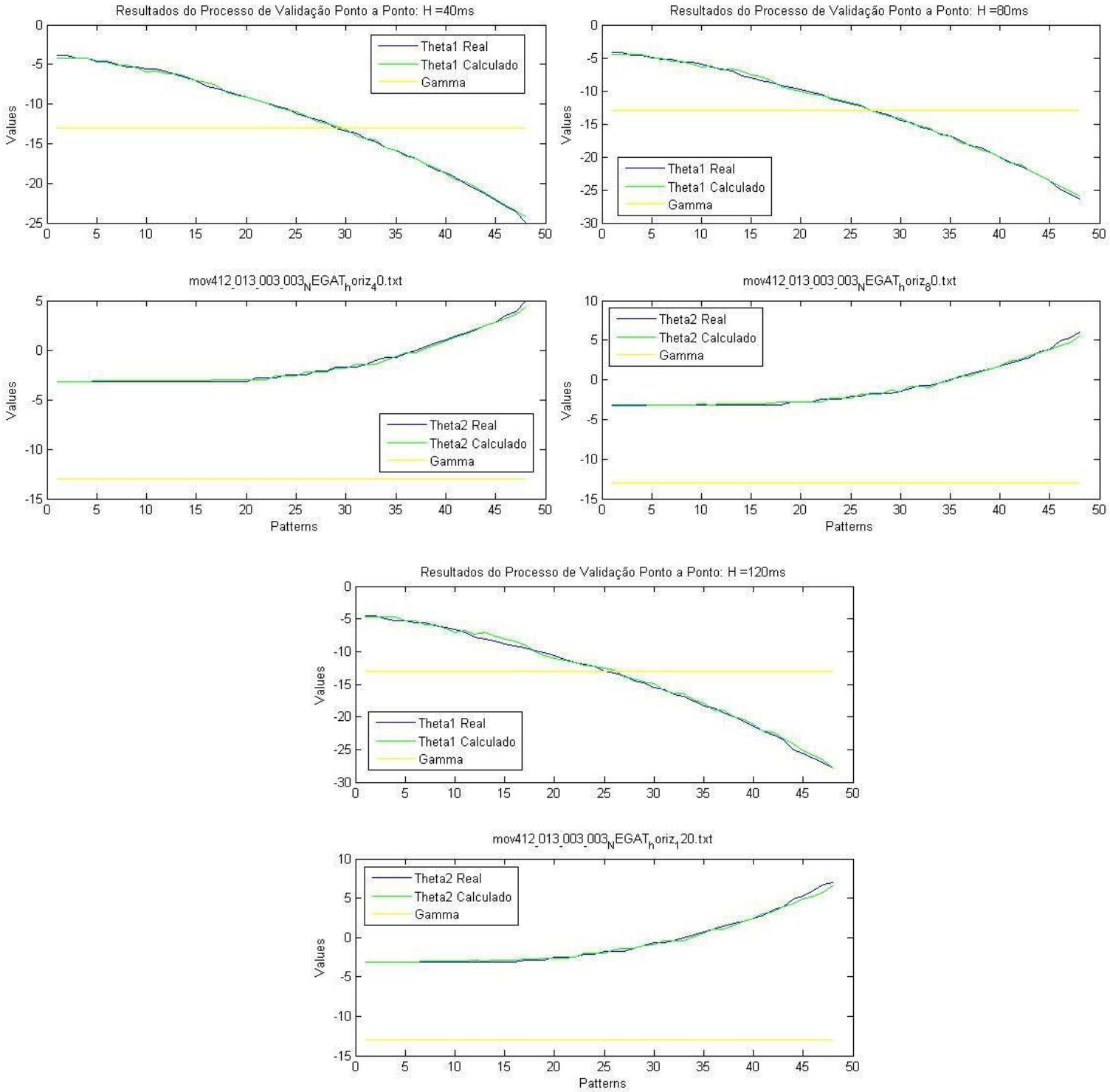
(d)

Trajetória 5: condição inicial ($\gamma = -3°, \theta_1 = 0°, \theta_2 = -7°$):



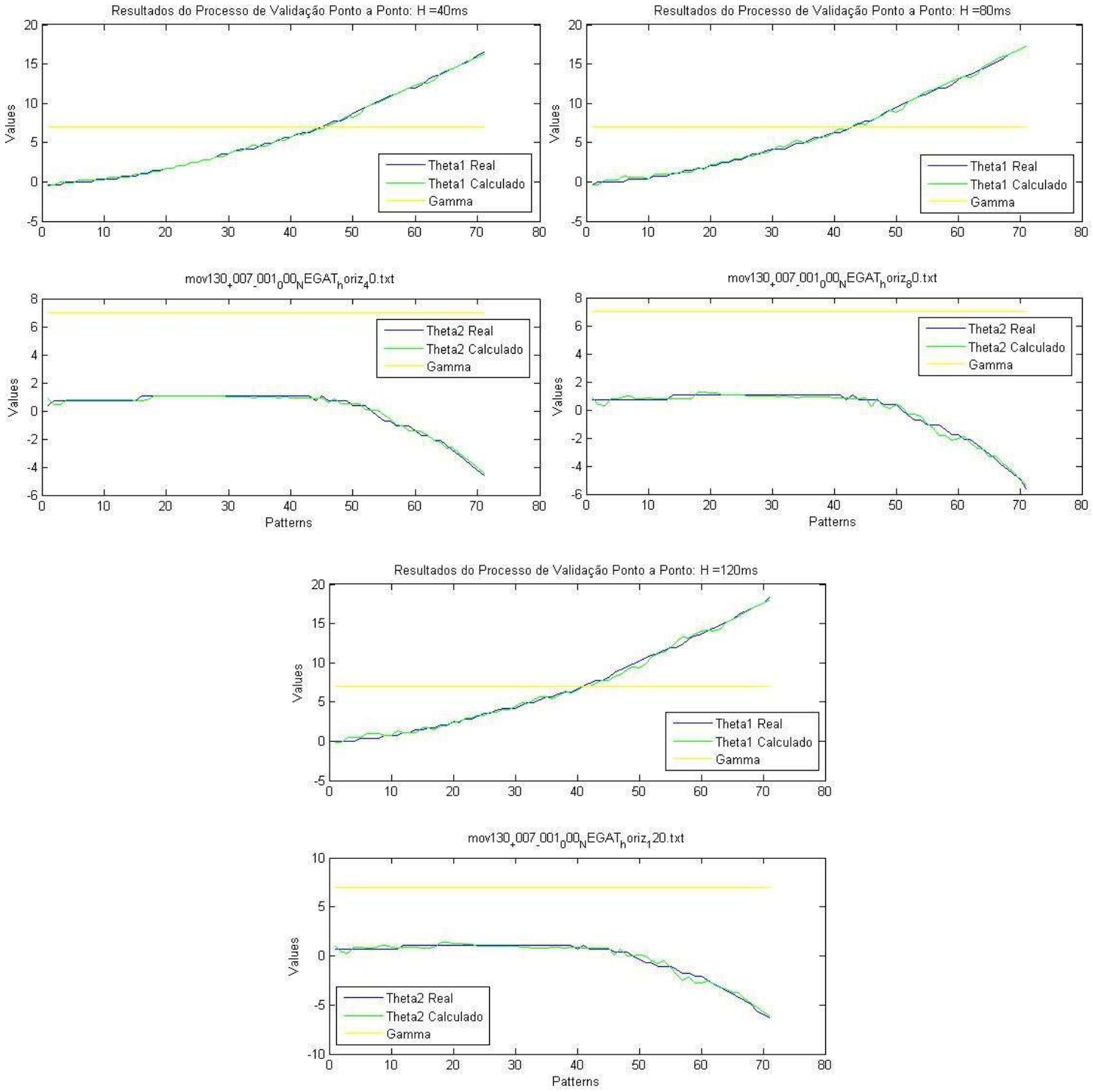
(e)

Trajetória 6: condição inicial ($\gamma = -13°, \theta_1 = -3°, \theta_2 = -3°$):



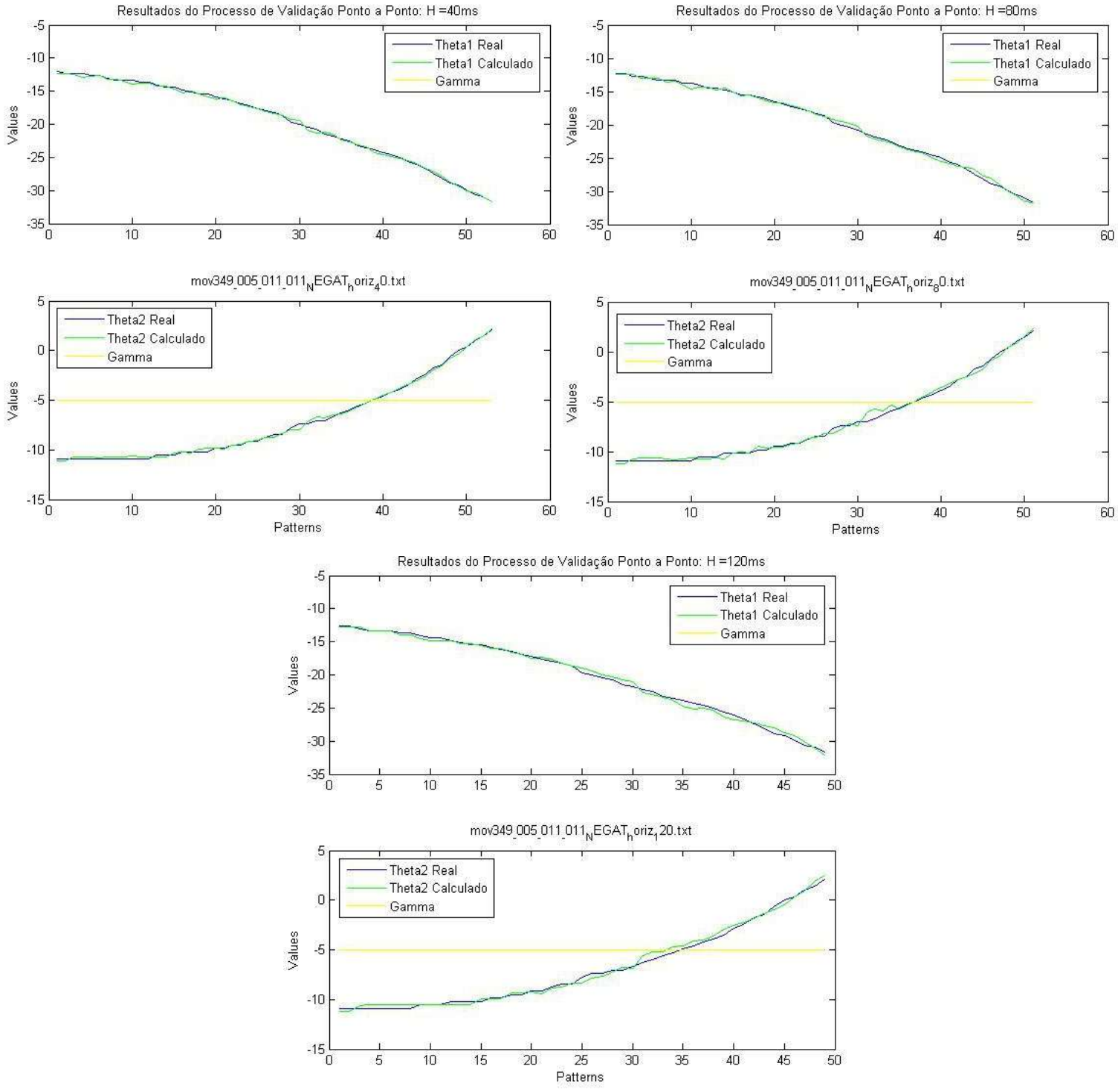
(f)

Trajetória 7: condição inicial ($\gamma = +7°, \theta_1 = -1°, \theta_2 = 0°$):



(g)

Trajetória 8: condição inicial ($\gamma = -5°, \theta_1 = -11°, \theta_2 = -11°$):



(h)

Figura 5.8 Alguns resultados do processo de Validação Ponto-a-Ponto para alguns preditores com horizontes H = 40*ms*, 80*ms* e 120*ms*.

Considerando todas as trajetórias testadas (inclusive as mostradas acima), pode-se concluir que todos os preditores conseguiram interpolar e, consequentemente, generalizar bem, seguindo cada trajetória, padrão a padrão.

As abordagens mostradas nas seções anteriores diferem na forma em que os dados destinados à validação são obtidos, porém ambas visam medir a capacidade de generalização da rede e ambas são similares na forma de interação do preditor com os dados. A seção a seguir mostra um caso especial de validação, ao qual os preditores foram submetidos.

5.4.3 Validação com Erro Cumulativo

O passo seguinte consistiu em reconstruir aproximações válidas de trajetórias convexas diversificadas (usadas no treinamento ou, de preferência, novas a serem coletadas), realizadas pelo veículo até o mesmo atingir *jackknife*, com o ângulo γ mantido constante. Porém, nesse caso, somente a condição inicial de cada trajetória era fornecida ao preditor, e não todos os padrões entrada/saída.

Cada preditor foi colocado, primeiramente, na mesma condição inicial de uma dada trajetória e, em seguida ele simulava, em malha aberta, o comportamento teórico do robô, utilizando, como novas entradas, os seus próprios dados de saída obtidos a cada intervalo H correspondente, ou seja, foram analisadas as respostas do preditor a entradas calculadas e fornecidas por ele mesmo em um intervalo de tempo correspondente ao seu horizonte de predição.

Esse procedimento foi efetuado durante o mesmo tempo de duração da trajetória correspondente. As curvas correspondentes a essas respostas tendem normalmente a divergir das originais coletadas, mas não devem deixar de reproduzir o comportamento adequado. Durante esse processo, denominado Validação com Erro Cumulativo, foram feitas algumas simulações utilizando horizontes de predição variados (iguais a 20*ms*, 40*ms*, 60*ms*, 80*ms*, 100*ms* e 120*ms*) e seus desempenhos foram comparados. O esquema da Figura 5.9 ilustra a etapa discutida.

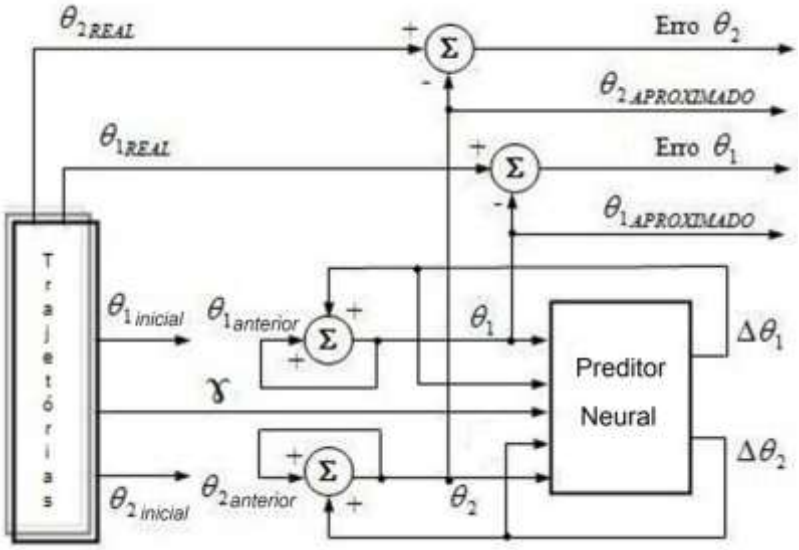
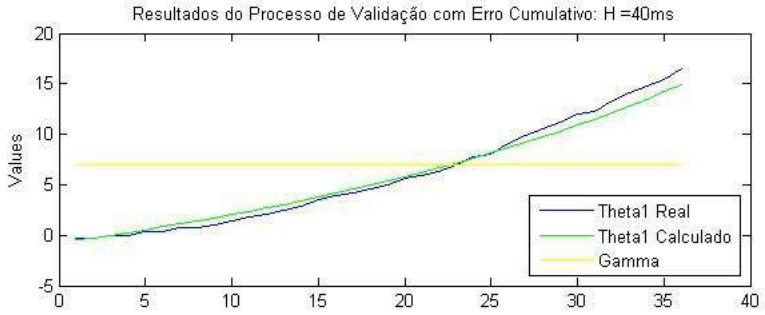
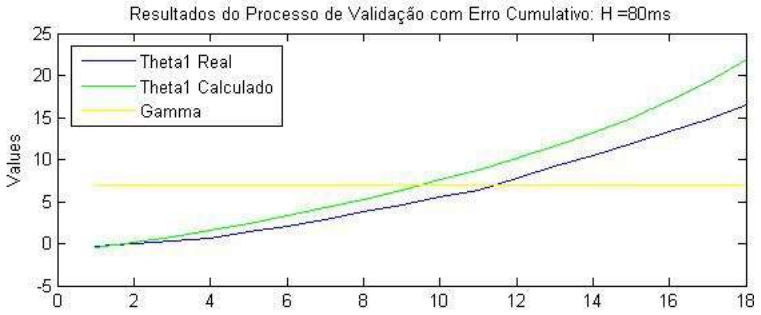


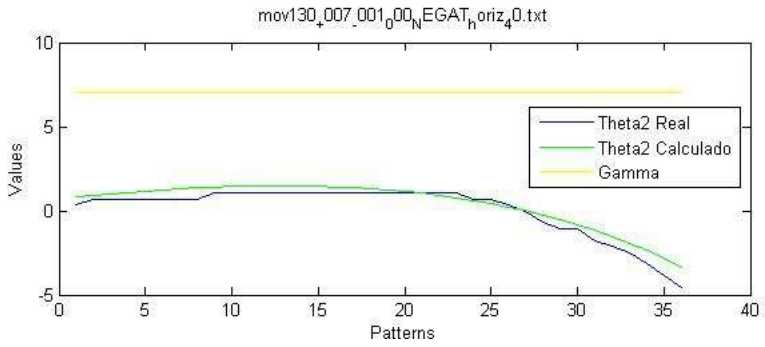
Figura 5.9 Esquema do processo de Validação com Erro Cumulativo de um preditor neural de horizonte H.

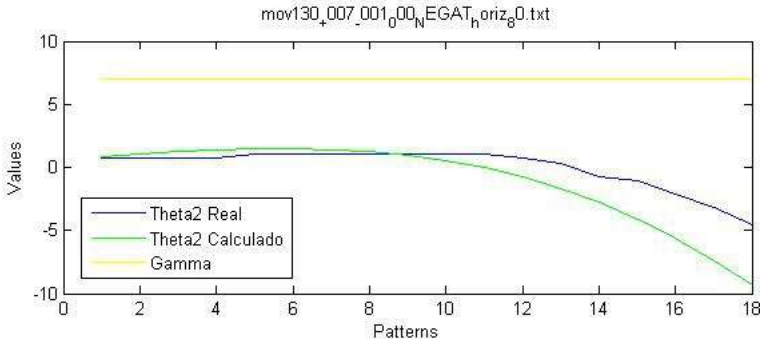
Alguns resultados do processo de Validação com Erro Cumulativo são apresentados na Figura 5.10. Nos gráficos, as entradas em amarelo correspondem aos valores do ângulo de direção do RMMA, também mantidos, nesse caso, constantes; as respostas em azul representam os valores dos ângulos correspondentes às trajetórias reais; e os valores em verde representam a resposta quando, dada uma condição inicial real $(\gamma_{inicial}, \theta_{1_{inicial}}, \theta_{2_{inicial}})$, um preditor calcula novos valores de ângulos a servirem de sua entrada nas próximas iterações. Para cada trajetória, o tempo decorrido é calculado multiplicando-se o número da iteração corrente (equivalente ao número do padrão atual) pelo horizonte de predição, H, correspondente. Obviamente, o tempo total de uma trajetória independe do horizonte empregado, ou seja, é o mesmo para todos os horizontes. No eixo vertical estão representados os ângulos em graus (*Values*), enquanto que o eixo horizontal corresponde à quantidade de padrões ou iterações (*Patterns*).

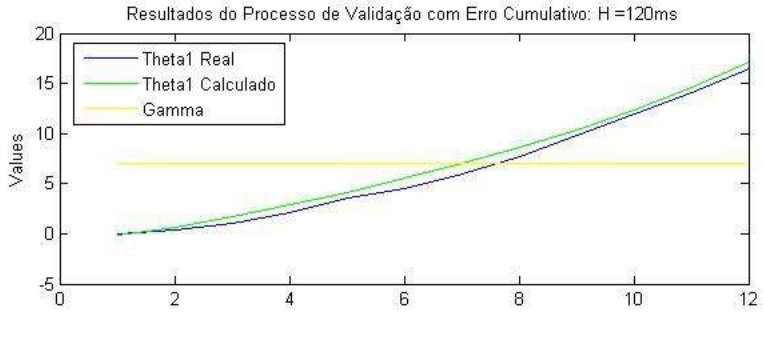
Trajetória 1: condição inicial ($\gamma = +7°, \theta_1 = -1°, \theta_2 = 0°$): *veja trajetória 7 em Validação Ponto-a-Ponto.*

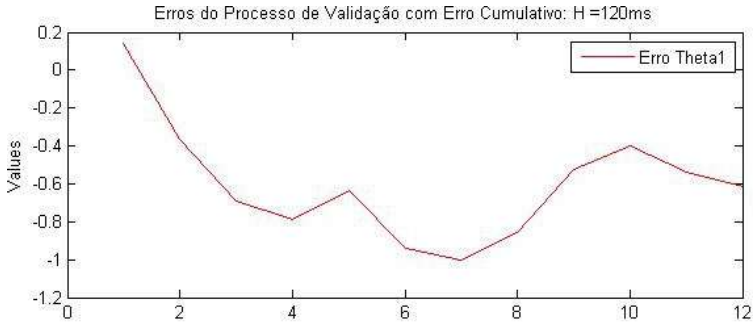


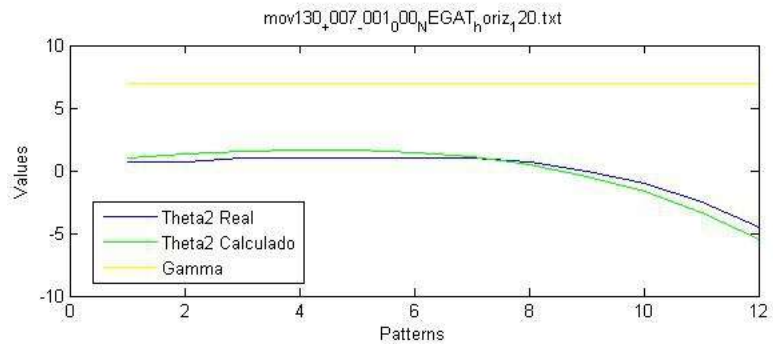


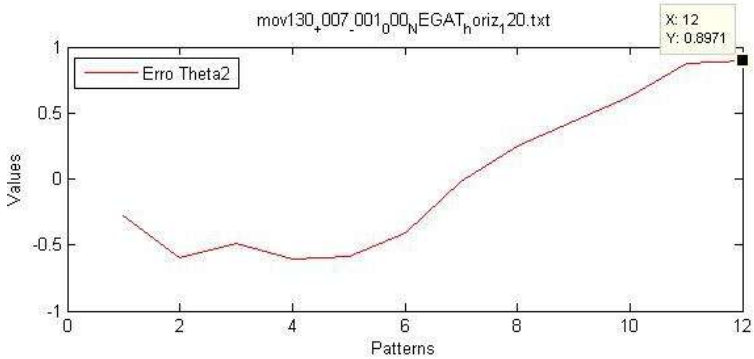






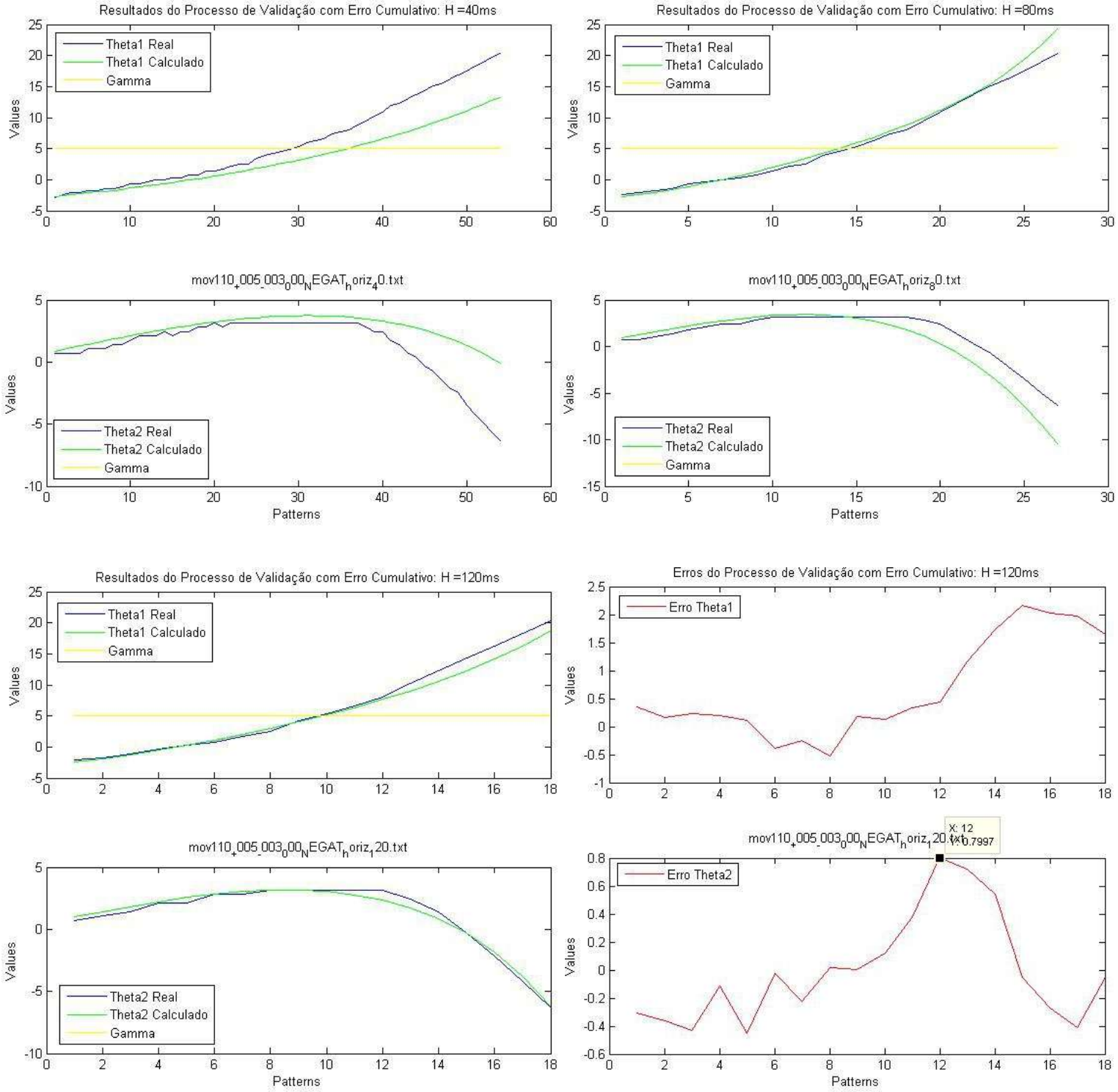






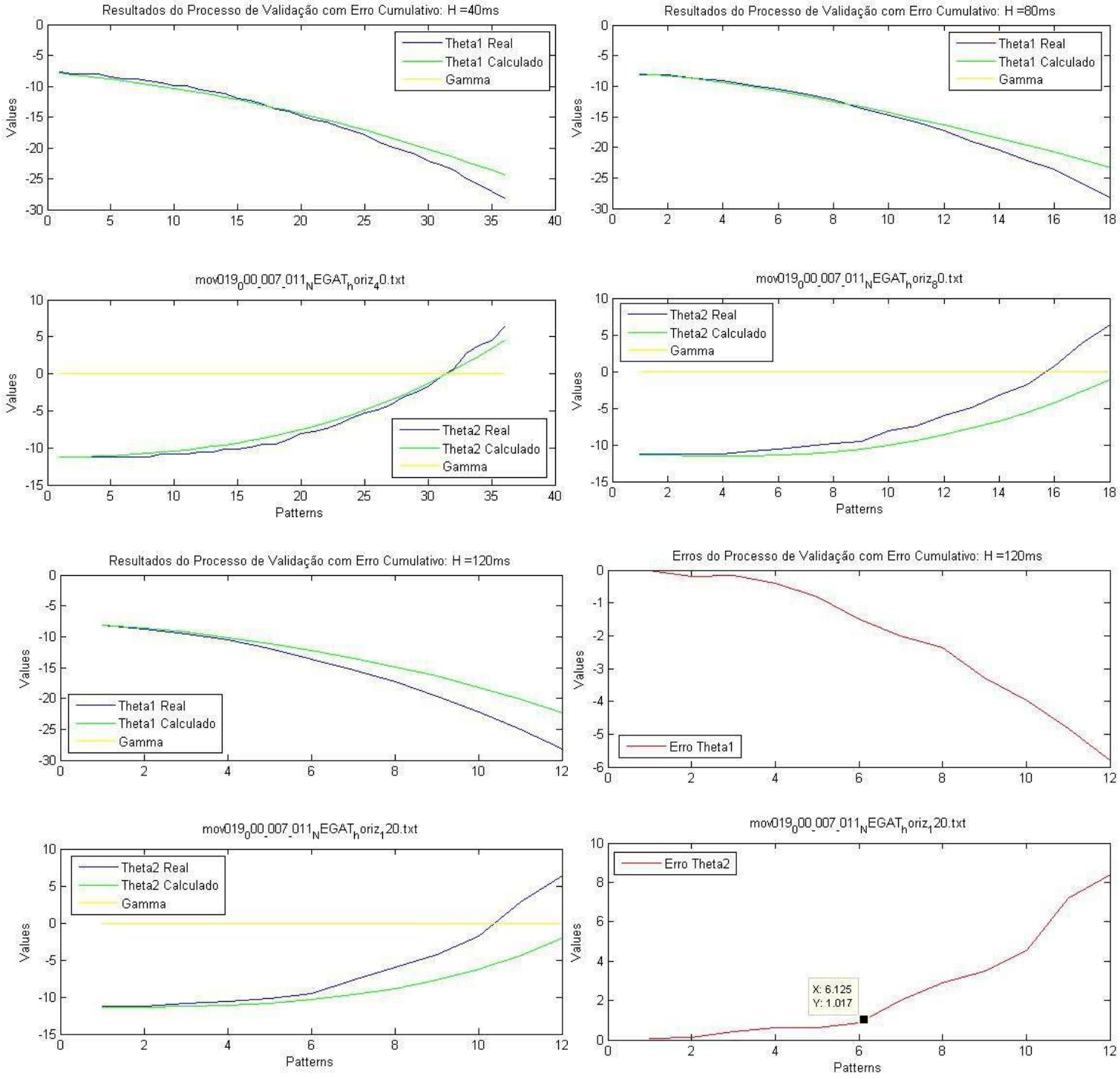
(a)

Trajetória 2: condição inicial ($\gamma = +5°, \theta_1 = -3°, \theta_2 = 0°$):



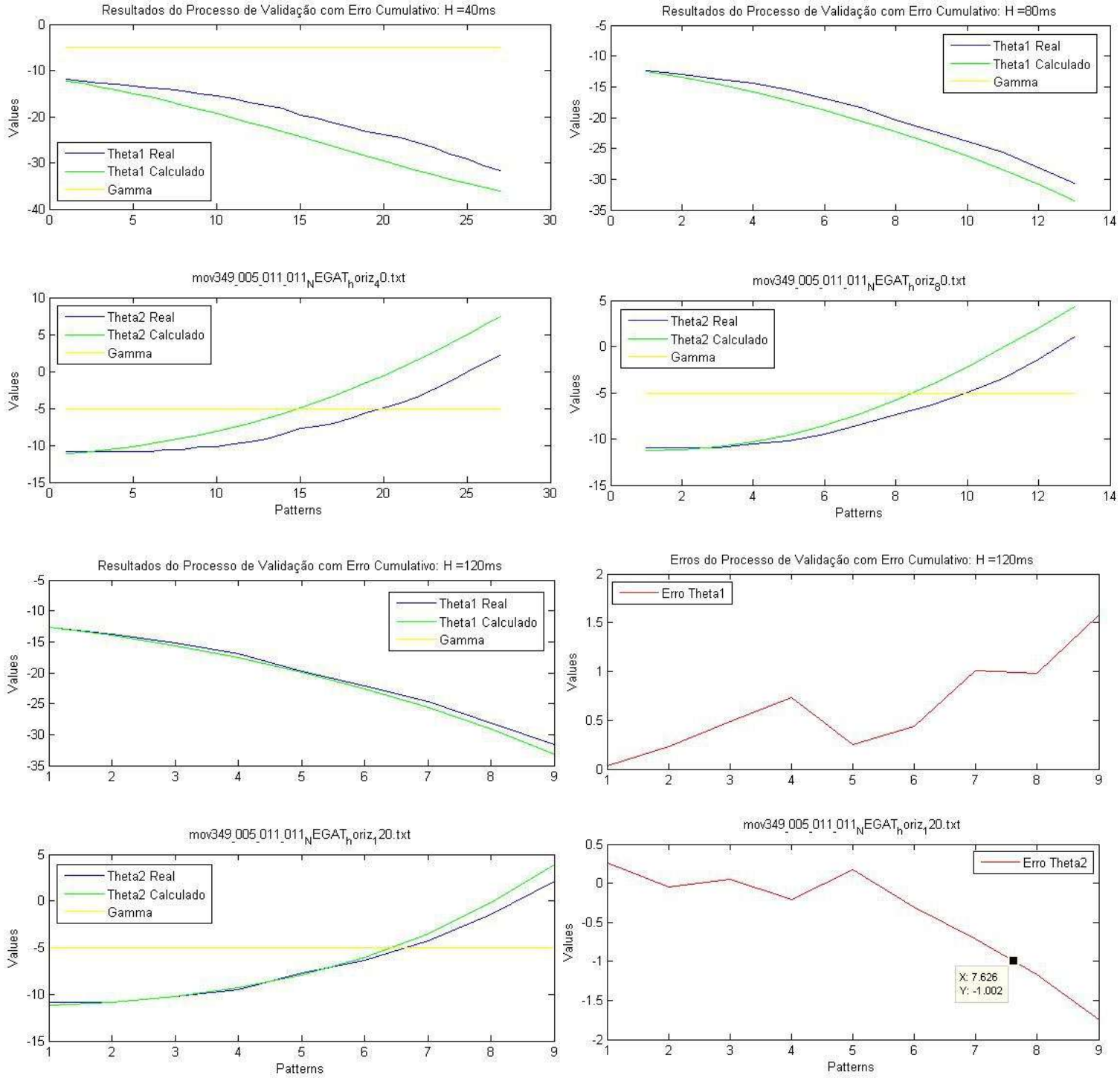
(b)

Trajetória 3: condição inicial ($\gamma = 0°, \theta_1 = -7°, \theta_2 = -11°$):



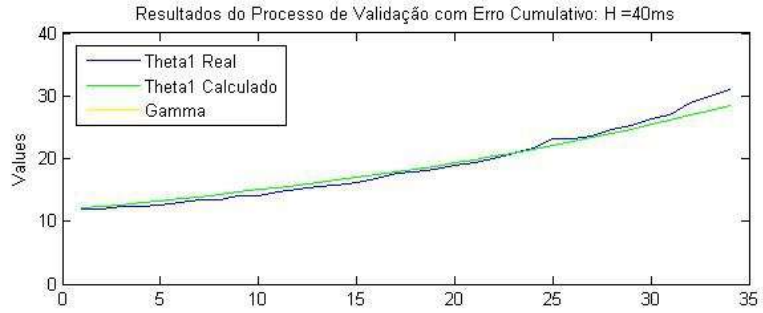
(c)

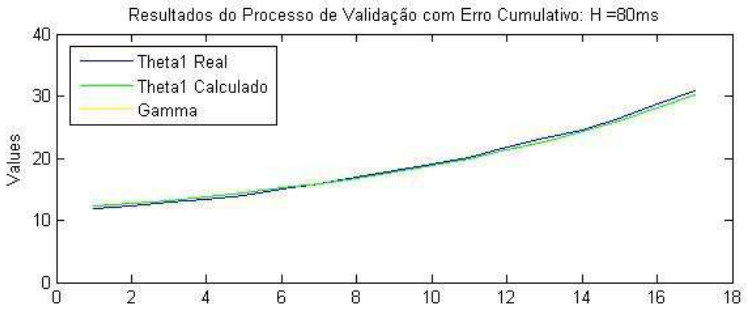
Trajetória 4: condição inicial ($\gamma = -5°, \theta_1 = -11°, \theta_2 = -11°$): *veja trajetória 8 em Validação Ponto-a-Ponto.*

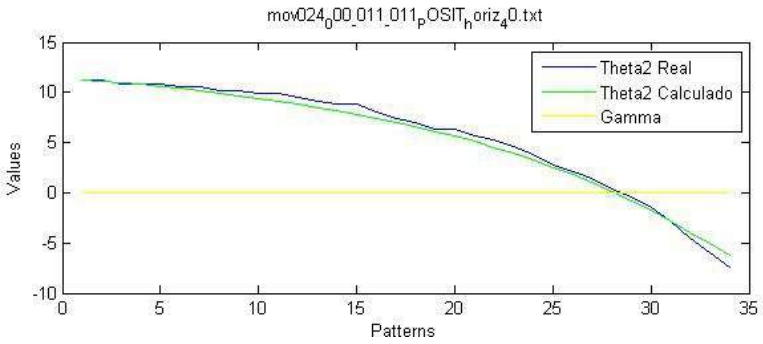


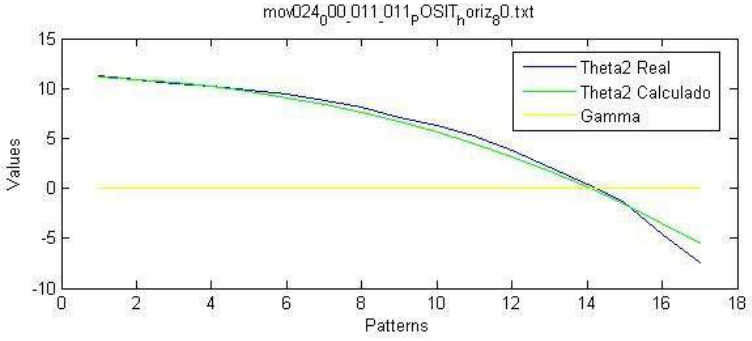
(d)

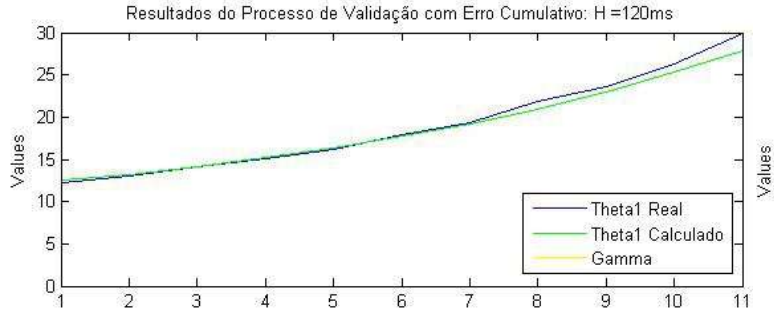
<u>Trajetória 5:</u> condição inicial ($\gamma = 0°, \theta_1 = +11°, \theta_2 = +11°$):

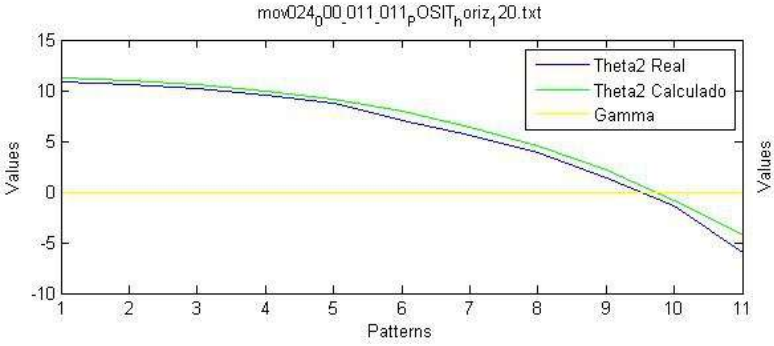


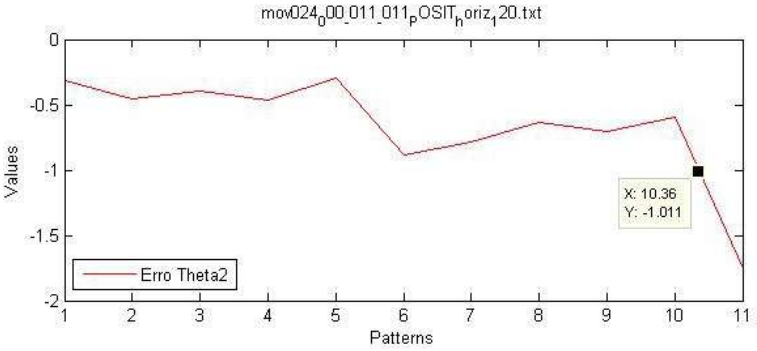






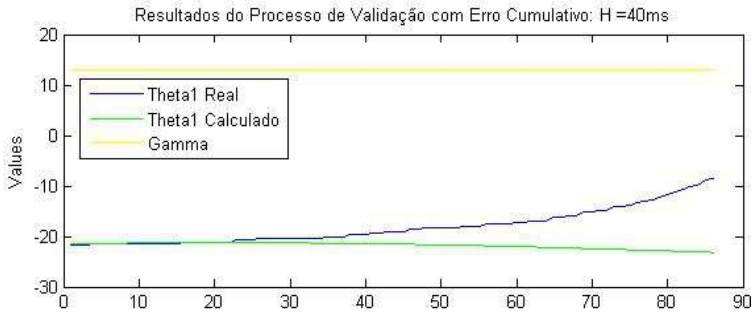


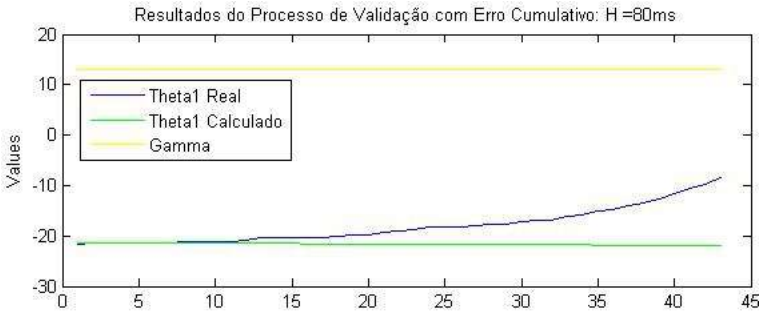


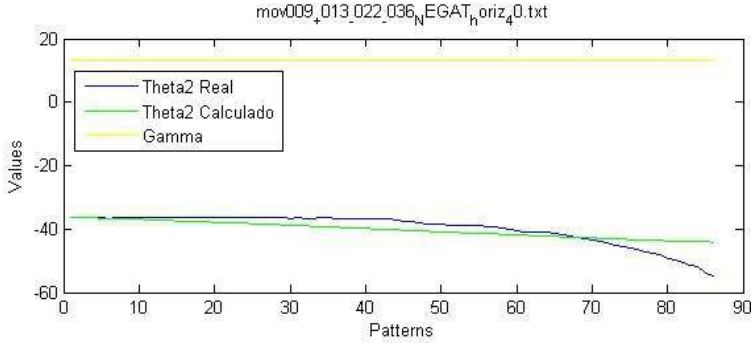


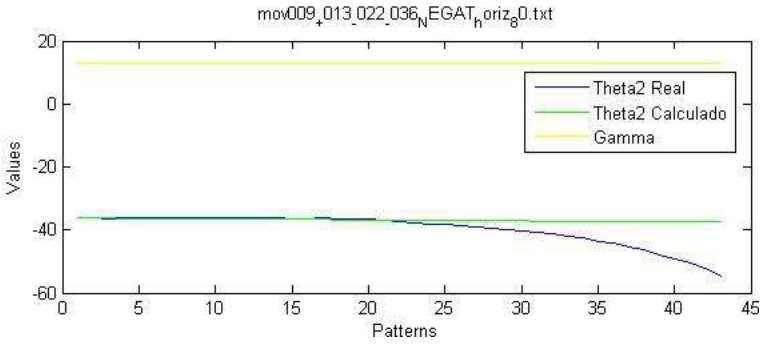
(e)

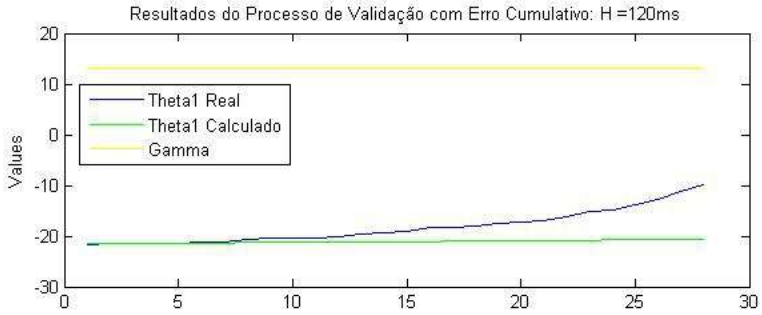
Trajetória 6: condição inicial ($\gamma = +13°, \theta_1 = -22°, \theta_2 = -36°$):

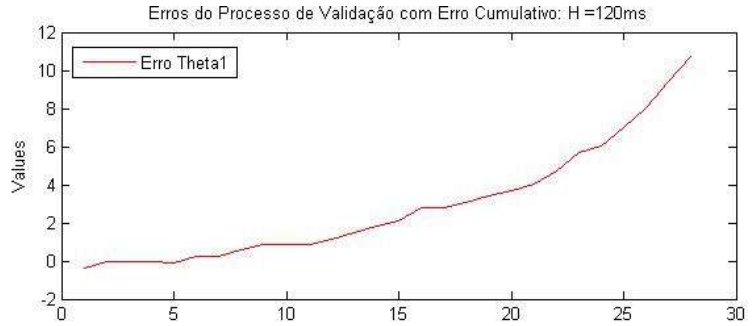


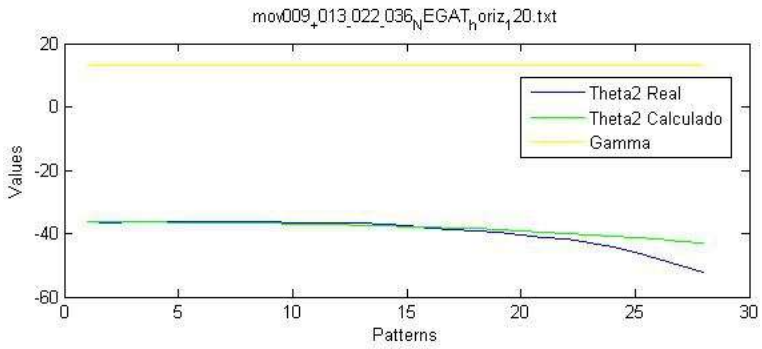


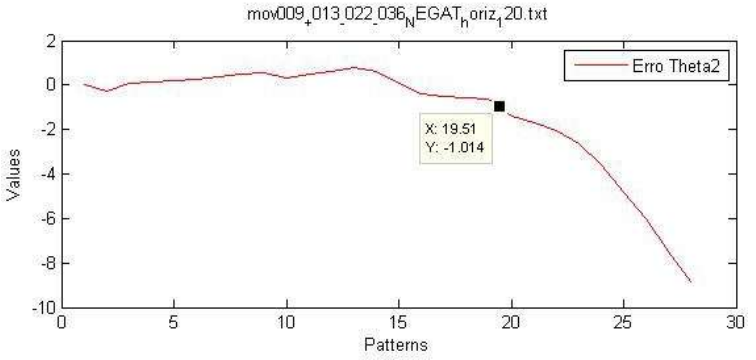






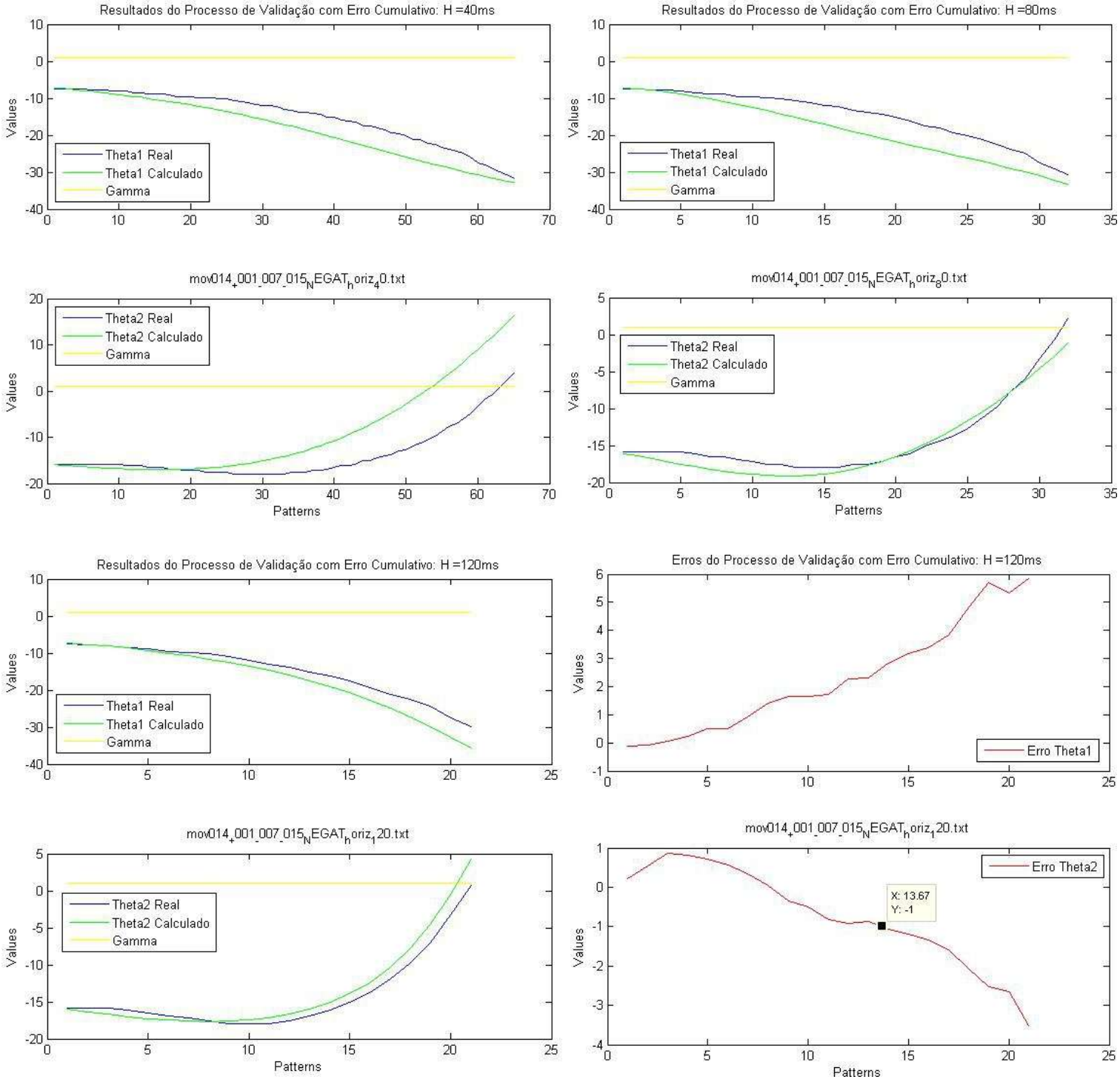






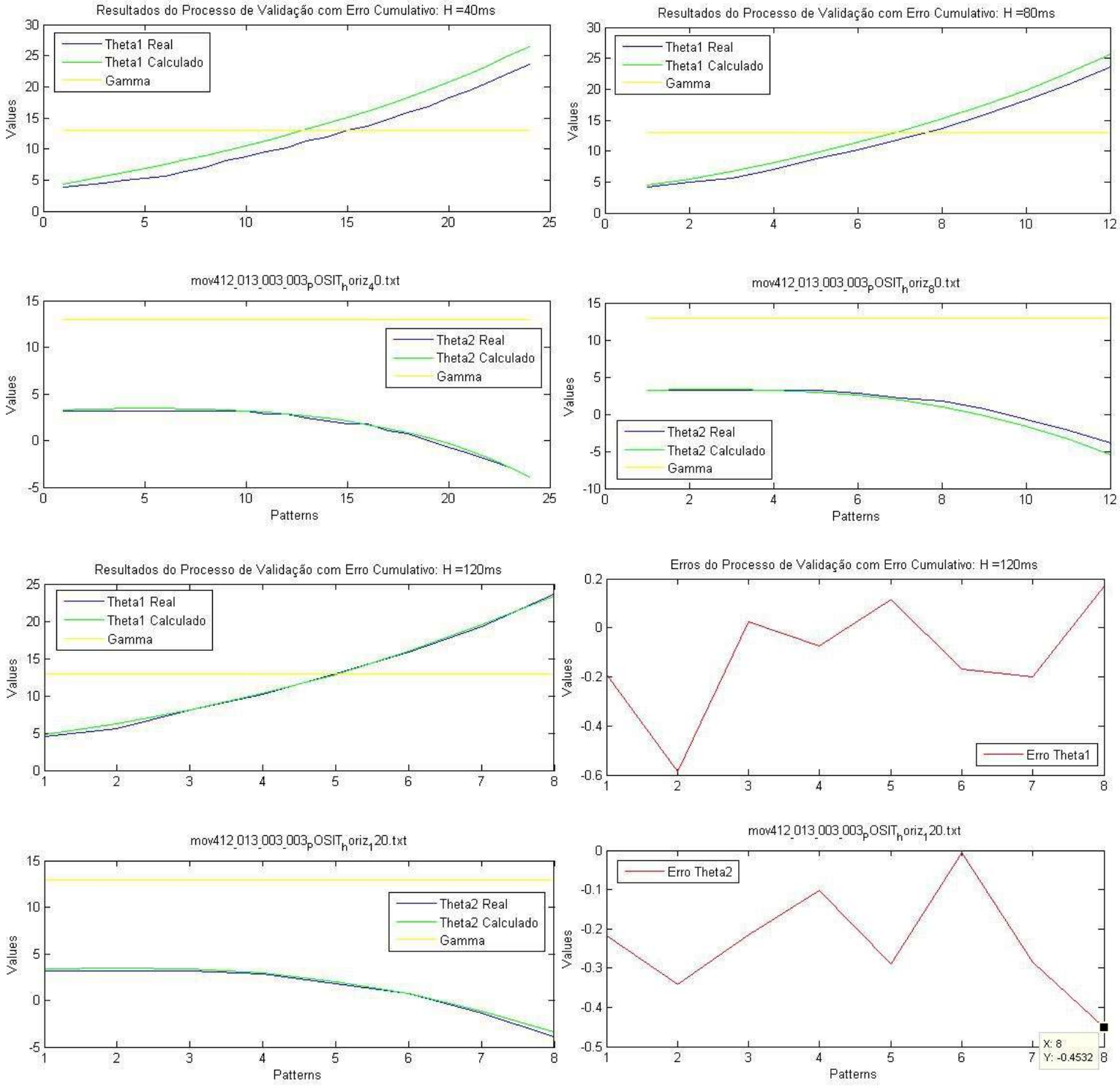
(f)

Trajetória 7: condição inicial ($\gamma = +1°, \theta_1 = -7°, \theta_2 = -15°$): *veja trajetória 3 em Validação Ponto-a-Ponto.*



(g)

Trajetória 8: condição inicial ($\gamma = +13°, \theta_1 = +3°, \theta_2 = +3°$):



(h)

Figura 5.10 Alguns resultados do processo de Validação com Erro Cumulativo para alguns preditores com horizontes H = 40*ms*, 80*ms* e 120*ms* (são mostrados os gráficos dos erros para esse último horizonte).

Nessa etapa, procura-se determinar o preditor que obteve o melhor desempenho relativo. Assim, se estabelece um horizonte de predição que definirá um modelo neural para o RMMA e que, consequentemente, servirá de base para a implementação do controlador. Além disso, tal preditor poderá ser usado em simulações com o controlador projetado ou em operações assistidas. Uma maneira de avaliar o desempenho de cada preditor, durante essa etapa, consiste em medir, para todas as trajetórias, a raiz quadrada do erro médio quadrático (*rmse*) entre as respostas simuladas e as reais. A maior quantidade de trajetórias com menor erro, em geral, define o horizonte H e o correspondente preditor final com melhor desempenho relativo.

Após alguns experimentos, tendo por base todas as trajetórias testadas, o preditor que demonstrou relativamente um melhor comportamento, em geral, com uma maior quantidade de trajetórias para as quais se obteve os menores *rmse's*, foi o de horizonte igual a 120*ms*. Tal fato pode ser melhor observado na Figura 5.11 que evidencia os *rmse's* obtidos por cada preditor em algumas das trajetórias descritas. A primeira coluna da tabela representa o horizonte de predição e a primeira linha, o número de cada trajetória. Dentro de cada célula, a primeira linha corresponde aos erros calculados nas trajetórias para o ângulo θ_1, já a segunda se refere aos erros calculados nas trajetórias para o ângulo θ_2. A seguir, a estratégia para a escolha do melhor preditor será explicada em maiores detalhes.

Horiz / Traj	1	2	3	4	5	6	7	8	9	10	11
40	2.1833	2.4340	1.5285	0.6921	9.3125	6.9192	4.2316	4.3056	1.9207	0.8798	1.9207
	6.0040	1.7203	1.5913	0.4756	11.7173	10.6925	1.7676	3.3330	0.9806	0.5422	0.9806
80	6.4698	3.6261	4.5995	2.4744	2.4387	2.6152	3.2140	1.9738	0.6149	0.3711	1.3669
	8.2264	3.6577	6.1180	2.0490	2.8142	2.4691	2.0455	1.9969	0.5533	0.6581	0.7455
120	2.8524	1.6018	1.5166	0.6693	1.4090	0.9266	2.7800	0.7861	1.6414	0.7550	0.2494
	5.9108	2.9693	4.1454	0.5612	7.2422	1.4098	1.6802	0.7586	0.4215	0.7689	0.2718

Configurações iniciais de cada trajetória:

Trajetória 1: $\gamma = +3°$, $\theta_1 = -1°$, $\theta_2 = 0°$ / Trajetória 2: $\gamma = +3°$, $\theta_1 = -3°$, $\theta_2 = -1°$ / Trajetória 3: $\gamma = +5°$, $\theta_1 = -3°$, $\theta_2 = -1°$ / Trajetória 4: $\gamma = +7°$, $\theta_1 = -1°$, $\theta_2 = 0°$ / Trajetória 5: $\gamma = +10°$, $\theta_1 = -11°$, $\theta_2 = -3°$ / Trajetória 6: $\gamma = +13°$, $\theta_1 = -11°$, $\theta_2 = 0°$ / Trajetória 7: $\gamma = -5°$, $\theta_1 = 0°$, $\theta_2 = -3°$ / Trajetória 8: $\gamma = -5°$, $\theta_1 = -11°$, $\theta_2 = -11°$ / Trajetória 9: $\gamma = -13°$, $\theta_1 = -3°$, $\theta_2 = -3°$ / Trajetória 10: $\gamma = 0°$, $\theta_1 = +11°$, $\theta_2 = +11°$ / Trajetória 11: $\gamma = +13°$, $\theta_1 = +3°$, $\theta_2 = +3°$

Número de trajetórias com menor erro para θ_1: 40*ms* = 1, 80*ms* = 2, **120*ms* = 8**

Número de trajetórias com menor erro para θ_2: 40ms = 4, 80ms = 1, **120ms = 6**

Figura 5.11 *Root mean square errors* (*rmse's*) para algumas trajetórias.

Para o caso mais geral, a Tabela 5.5, a seguir, resume o comportamento dos preditores em todas as trajetórias testadas. Conforme comentado no início da presente seção, esse conjunto diversificado de trajetórias é formado pelos movimentos convexos usados na constituição das bases de treinamento e por alguns novos movimentos coletados. Essa tabela se assemelha muito à Tabela 5.2, já que o processo empregado de validação e a estratégia destinada a medir o desempenho de cada rede são os mesmos em ambos os casos, porém,

nesse caso, se trabalha com a estrutura neural definida anteriormente aplicada à preditores com horizontes de predição, H, diferentes.

Tabela 5.5 Desempenho de cada preditor com horizonte diferente no processo de Validação com Erro Cumulativo.

Preditores neurais (coluna 0)	**Quantidade de trajetórias onde o preditor obteve menor erro (*rmse*) para θ_1**	**Quantidade de trajetórias onde o preditor obteve menor erro (*rmse*) para θ_2**	**Média dos menores erros (*rmse's*) por trajetória para θ_1**	**Média dos menores erros (*rmse's*) por trajetória para θ_2**
H = 40*ms*	*294*	*314*	*1.8717*	*1.7234*
H = 80*ms*	*292*	*291*	*2.0689*	*2.2285*
H = 120*ms*	***364***	***345***	*2.1983*	*1.9695*

Inicialmente, foram calculados os *rmse's* em cada trajetória, para cada preditor. Em seguida, foi identificado, para cada trajetória, o preditor que obteve o menor valor de erro (*rmse*). Por fim, foi possível determinar, para um determinado preditor: o conjunto formado pelas trajetórias, nas quais o mesmo desempenhou melhor comportamento, ou seja, trajetórias, nas quais os erros (*rmse's*) obtidos pelo preditor foram os menores quando comparados aos erros obtidos pelos outros preditores (tamanho desse conjunto – colunas 1 e 2 da Tabela 5.5); e o conjunto formado pelos valores desses erros, considerados os menores por trajetória (média desse conjunto - colunas 3 e 4 da Tabela 5.5). Esses procedimentos foram feitos para os ângulos θ_1 e θ_2. Com isso, pôde-se definir o preditor com melhor desempenho considerando ambos os ângulos.

O critério de escolha do preditor com melhor desempenho foi baseado nos resultados referentes às duas primeiras colunas da Tabela 5.5. O preditor escolhido seria aquele com as maiores quantidades de trajetórias com menores erros, considerando ambos os ângulos, simultaneamente. Caso diferentes preditores demonstrassem melhores resultados para os diferentes ângulos, aquele com a maior soma da quantidade de trajetórias com menores erros, para ambos os ângulos, seria o escolhido.

Logo, ao se observar a Tabela 5.5, pôde-se confirmar que o preditor com melhor desempenho foi o de horizonte de predição de 120*ms*, como mencionado antes na Figura 5.11. Para todos os preditores, os menores erros obtidos em cada trajetória, para θ_1 e θ_2, não possuíam valores considerados altos, e sim valores dentro de limiares aceitáveis. Esse fato pode ser confirmado a partir dos resultados evidenciados nas duas últimas colunas da tabela, os quais representam as médias dos menores erros obtidos em cada trajetória por cada

preditor. Isso também ocorreu quando da escolha da estrutura neural, como mostrado na Tabela 5.2.

A Figura 5.12 mostra a parte da *Interface* implementada responsável pelos processos detalhados de validação dos preditores com horizontes de predição variados.

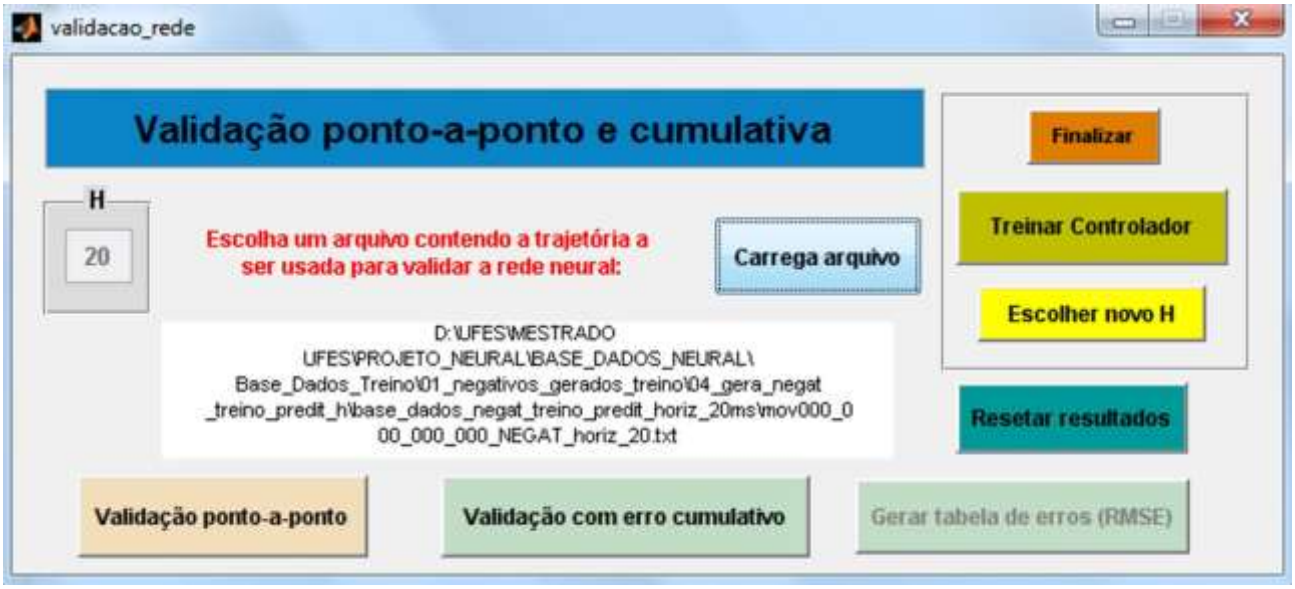


Figura 5.12 Parte da *Interface* destinada às etapas de Validação Ponto-a-Ponto e de Validação com Erro Cumulativo (*validacao_rede.m, .fig*).

5.4.4 Determinação do Horizonte de Validade do Preditor

Pela análise das trajetórias reproduzidas pelo preditor de horizonte de 120*ms*, durante a fase de Validação com Erro Cumulativo, pôde-se observar que, de uma maneira geral, o comportamento do mesmo, foi satisfatório conseguindo, em malha aberta, dada uma condição inicial, reproduzir com um grau aceitável de precisão a evolução dos ângulos de configuração medidos a partir da movimentação do robô.

Apesar dos bons resultados anteriormente apresentados, para efeitos de validação, o preditor neural, de qualquer forma, não foi concebido para ser utilizado em malha aberta e muito menos com números de iterações longos quanto os apresentados na fase de validação da rede. Deve-se observar que, normalmente, quanto menor o número de iterações (representado por *patterns* no eixo horizontal dos gráficos da Seção 5.4.3) e, consequentemente, quanto menor o tempo decorrido, o erro entre o modelo neural e o modelo real torna-se reduzido; à medida que o tempo decorrido aumenta, a curva aproximada de cada componente angular tende a divergir de sua respectiva curva real, devido ao fato de tal erro ser cumulativo, conforme já era de se esperar. Logo, faz-se necessário delimitar o universo de

validade do preditor obtido, segundo um intervalo de tempo múltiplo do seu horizonte de predição, atrelado a uma margem de erro aceitável.

Em outras palavras, uma vez definido um preditor que obteve um desempenho relativo satisfatório, se procura obter um horizonte válido que limita a sua atuação em simulações ao substituir a planta original, o que determina o tempo máximo de simulação com o referido preditor em operação assistida ou em experimentos em malha fechada de controle usando qualquer tipo de controlador (nesse último caso, pode-se dizer que esse horizonte representa o intervalo de tempo dentro do qual deve ser tomada a ação correta de controle). Para isso, se faz necessário identificar o limite de tempo até onde o erro absoluto entre a resposta estimada pelo preditor e a correspondente trajetória real está dentro de um limiar aceitável; o menor valor de tempo ("estratégia do pior caso"), considerando todas as trajetórias, constitui o horizonte de validade do preditor em simulações.

Por outro lado, outra estratégia, que não considera o menor valor de limite de tempo em todas as trajetórias, também pode ser empregada na determinação do horizonte de validade do preditor. Nessa, as trajetórias descritas são agrupadas conforme o desempenho do preditor, que é avaliado levando-se em consideração a relação entre o número de padrões (ou iterações) atingidos até se alcançar o erro estabelecido e o tamanho total da respectiva trajetória. O horizonte de validade do preditor é computado por meio de uma média ponderada, onde a quantidade de trajetórias em cada grupo é usada como peso dessa ponderação.

Para efeitos de simplificação de esforço, as trajetórias mostradas na Figura 5.10 são usadas para o cálculo do horizonte de validade do preditor de $120ms$. Assumindo uma margem de erro aceitável de aproximadamente 1° para o último ângulo de configuração, θ_2, a partir dos resultados apresentados na Tabela 5.6, conclui-se que:

- três (1, 2 e 8) das oito respostas geradas pelo preditor apresentam comportamento mais satisfatório, com valor de erro máximo, ao longo da trajetória, menor que o estabelecido – Grupo 1;
- em uma das respostas (3), a quantidade de padrões para se atingir o erro estabelecido equivale a 50% do tamanho da trajetória, mesmo assim, essa quantidade é pequena em relação aos demais resultados – Grupo 2;
- em duas respostas (4 e 5), a quantidade referida acima fica a um padrão de diferença do número total de padrões da trajetória – Grupo 3;
- por fim, em duas respostas (6 e 7), a quantidade referida é maior que 50% do tamanho da trajetória e possui valor relativamente maior que no caso do grupo 3 – Grupo 4.

Tabela 5.6 Dados obtidos a partir de trajetórias descritas para se determinar o horizonte de validade de um preditor.

Trajetória (resposta) descrita pelo preditor	Tamanho total da trajetória (em padrões)	Número de padrões para se atingir o erro estabelecido	Maior erro obtido, dentro do valor estabelecido	Grupos	Número de padrões correspondentes a cada grupo
1	12	12	0.8971	1	18
2	18	12	0.7997	1	18
3	12	6	1.017	2	6
4	9	8	-1.002	3	8
5	11	10	-1.011	3	8
6	28	20	-1.014	4	14
7	21	14	-1	4	14
8	8	8	-0.4532	1	18

Logo, o horizonte de validade do preditor de 120*ms* é calculado da seguinte maneira:

$$\Gamma = \frac{\boldsymbol{quantidade\ de\ trajetórias\ em\ cada\ grupo} \cdot (\boldsymbol{número\ de\ padrões\ de\ cada\ grupo})}{\boldsymbol{quantidade\ total\ de\ trajetórias\ descritas}}$$

$$\Gamma_{patterns} = \frac{1(6) + 2(8) + 2(14) + 3(18)}{8} = 13\ padrões$$

Sabendo que a cada iteração (padrão) decorrem 120*ms*, para o preditor correspondente, tem-se que o seu horizonte de validade, Γ, é de:

$$\Gamma_{ms} = 13\ padrões \times 120 \frac{ms}{padrão} = 1.56\ segundos$$

Considerando o tamanho (ou tempo) médio de todas as trajetórias descritas na etapa de Validação com Erro Cumulativo, de 15 padrões (ou de 15 x 120*ms* = 1.8 segundos), percebe-se que o horizonte de validade obtido para o preditor de 120*ms* é satisfatório para ser usado em simulações em malha aberta.

5.5 Estrutura Geral da *Graphical User Interface*

O esquema da Figura 5.13 resume os procedimentos implementados na *Interface*, desde a etapa de coleta de dados, passando pelo treinamento de um preditor e controlador com certo horizonte de predição até suas validações.

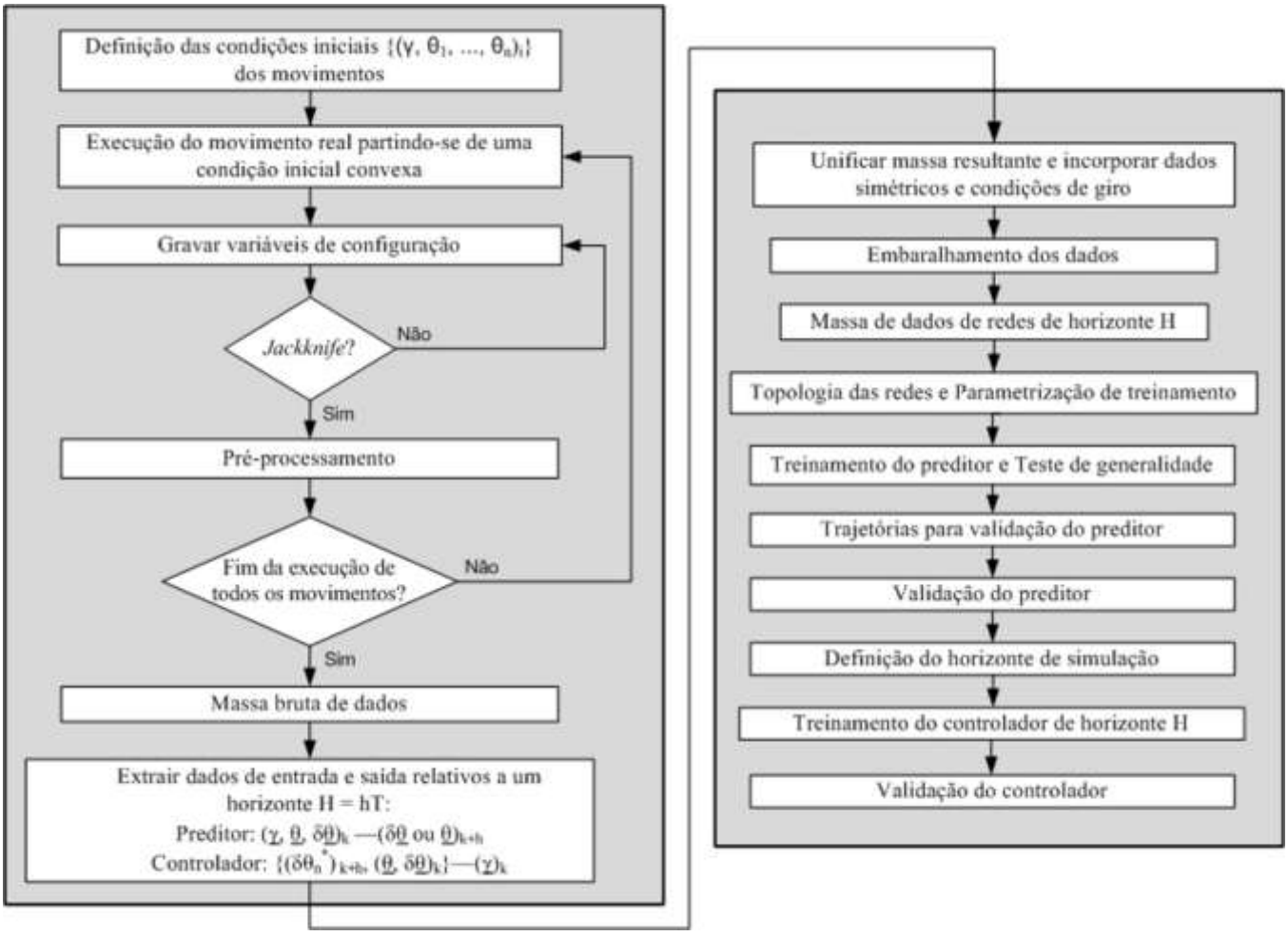


Figura 5.13 Estrutura da *Interface* implementada.

Finalmente, no próximo capítulo, serão abordados os aspectos referentes ao problema de controlar um RMMA efetuando movimentos à ré, constando desta análise a dissociação dos níveis de controle e supervisão, o projeto de controladores completos a horizonte fixo e a validação dos mesmos, em malha fechada, utilizando preditores neurais e analíticos.

Capítulo 6: O Problema de Controle

Antes de se discutir o tratamento ao problema de controle, será apresentada uma breve revisão dos principais componentes da arquitetura de controle mostrada no Capítulo 2 e dos tipos de movimentos em problemas de manobras e navegação de RMMA's, de modo a melhor situar o trabalho quanto ao objetivo do controlador nesse contexto.

6.1 Introdução

Sob o ponto de vista de controle, a variável de referência básica, obtida pela solução do problema de coordenação, que caracteriza o movimento à ré do RMMA no espaço de configuração, é o valor desejado do ângulo entre os dois últimos *trailers* da cadeia mecânica articulada, correspondendo à última componente, θ_n^*, do vetor de configuração, (ou sua variação angular, $\delta\theta_n^*$, considerando o horizonte de predição). Já, a variável de controle, γ, é o ângulo de direção das rodas dianteiras do robô.

As variáveis de configuração correspondem aos ângulos entre os elementos da composição, ou seja, aos ângulos de juntas (ângulo entre o *truck* e o primeiro *trailer* e ângulo entre os *trailers*). A quantidade dessas variáveis define o grau de liberdade do sistema. Situações em que os ângulos de configuração possuem os mesmos sinais numéricos representam configurações convexas. Configurações convexas particulares, correspondentes a movimentos circulares, com um correspondente raio de giro constante, são denominadas configurações em arco ou em giro. Já, configurações onde os ângulos de juntas possuem sinais distintos são chamadas côncavas; e, por fim, configurações, nas quais os valores de todos os ângulos são nulos, são denominadas alinhadas.

A implementação neural de um controlador completo a horizonte finito H, via rede estática, é realizada pelo seguinte mapeamento:

$$\left\{(\delta\theta_n^*)_{k+h}, \left(\underline{\theta}, \delta\underline{\theta}\right)_k\right\} - \left(\underline{\gamma}\right)_k \qquad (6.1),$$

onde o sobrescrito "*" indica a referência do sistema.

6.2 A Proposta de Controle

Devido às limitações inerentes ao próprio RMMA e à análise das equações de movimento da cadeia cinemática, mostradas no Capítulo 4, a probabilidade de o problema de controle neural ter solução para as configurações inicialmente convexas é relativamente maior. Assim, se a configuração inicial é côncava, é necessária uma etapa inicial de manobra para levar a composição para uma configuração alinhada ou convexa. Essa manobra inicial, em geral, aumenta os ângulos de configuração e pode levar a cadeia a uma situação crítica ou sem solução. Nesse caso, deve-se mover a cadeia para frente, diminuindo-se a configuração até que a solução exista. De qualquer forma, mesmo se a configuração convexa obtida é não crítica, a manobra inicial a ser realizada leva certo tempo, o que constitui um "atraso" em função da configuração, no controle de movimentos.

É muito difícil que uma solução genérica resolva, de modo implícito e com desempenho adequado, os problemas onde a configuração inicial é côncava ou problemas com "atraso de configuração". Desse modo, soluções customizadas são desenvolvidas para a manobra inicial. O controle do movimento à ré partindo o RMMA de configurações alinhadas pode ser desenvolvido por meio de regras *fuzzy* ou heurísticas *ad-hoc*, porém foge ao escopo do presente trabalho.

Análises circunstanciadas e ensaios em protótipos, buscando soluções que apresentem desempenho adequado em qualquer configuração, levaram, sem perda da generalidade, à conclusão de que uma abordagem adequada, alternativa à busca de soluções genéricas, é a de quebrar o problema de controle em classes distintas de problemas, em função da configuração da cadeia articulada. Cada classe, com uma simplificação correspondente, irá levar a uma solução personalizada e, finalmente, a uma solução global obtida pela comutação do controle entre classes. De qualquer forma, é de extrema utilidade quebrar o problema geral em classes, pois qualquer solução genérica deve ter o seu desempenho avaliado ou comparado com outras soluções, para as diversas classes de problemas. As classes de problemas são discriminadas em função da configuração inicial. Vale ressaltar que, em todos os casos, o objetivo de controle consiste em seguir uma dada referência para o ângulo (ou sua variação) entre as últimas articulações. A seguir são mostradas as classes de problemas, em ordem crescente de complexidade:

A. Transição entre movimentos em arco, com centro de giro do mesmo lado. Deve-se testar o controlador para amplitudes crescentes (em módulo) da referência de variação de giro;
B. *Tracking* (seguimento), partindo-se inicialmente o RMMA de uma configuração alinhada para uma de giro. Deve-se testar o controlador para amplitudes crescentes (em módulo) da referência de variação de giro;
C. *Tracking* (seguimento), partindo-se inicialmente o RMMA de uma configuração convexa qualquer para uma de giro, com centro de mesmo lado. Nesse caso, pode-se partir de uma configuração de giro com o intuito de mantê-la ao longo do movimento;
D. *Tracking* (seguimento), partindo-se inicialmente o RMMA de uma configuração convexa qualquer para uma de giro, com centro de lado oposto.

Nesses dois últimos casos, a ideia básica que deve nortear o desenvolvimento de uma estratégia de controle de bom desempenho é a de considerar, num intervalo de tempo pequeno, o controle de movimentos convexos aproximadamente como o controle da transição entre giros constantes "próximos".

Considerando os fatores mencionados anteriormente, é objeto de estudo deste trabalho o controle neural de movimentos à ré em configurações convexas. Especificamente, é abordado o comportamento do RMMA em movimentos correspondentes aos itens A, B e C acima.

6.3 Formas de Obtenção do Controlador Neural

O corrente trabalho explicita duas formas de obtenção dos controladores neurais: diretamente a partir dos dados coletados e a partir daqueles gerados pelo modelo de singularidades ou indiretamente a partir da planta, e em ambos os casos atuando como uma inversa aproximada (à esquerda) do preditor de horizonte H.

6.3.1 Controlador Indireto

De posse de um modelo neural de horizonte H da planta a ser controlada, pode-se, então, iniciar o processo de treinamento de um controlador a partir do mesmo. Para o

aprendizado desse tipo de controlador, alguns cuidados são necessários. Isso se dá, através do emprego do Matlab®, conforme alguns aspectos implementáveis, citados a seguir.

Primeiramente, a rede do preditor projetado é acoplada à direita de outra rede correspondente ao controlador, gerando uma rede formada pela concatenação de ambas. Em termos práticos de implementação, para efeitos de treinamento, a rede concatenada é vista como uma única rede (diferentemente de duas redes independentes ligadas em série), porém contendo parcelas estruturais relacionadas ao preditor previamente treinado e ao controlador a ser projetado. Suas entradas correspondem às entradas do controlador, enquanto que suas saídas correspondem às saídas do preditor.

Devem ser carregados nas camadas, da rede concatenada, destinadas a abrigar o preditor todos os parâmetros da rede previamente treinada como modelo neural, como por exemplo, seus pesos, seus *biases* e suas funções de ativação, enquanto que as camadas correspondentes ao controlador são criadas e parametrizadas da maneira que o usuário julgar mais adequada, por meio da *Interface* implementada.

Em seguida, o preditor é empregado de modo a propagar para trás, através de sua estrutura de pesos e *biases*, o erro a ser utilizado no ajuste dos parâmetros do controlador. É fundamental garantir a possibilidade de efetuar tal propagação sem, contudo, que a mesma modifique a estrutura de pesos do preditor previamente treinado. Logo, os parâmetros da parcela da rede concatenada correspondente ao preditor treinado não devem ser novamente modificados, durante o treinamento. Esse artifício pode ser implementado anulando-se a taxa de aprendizado para as camadas da rede concatenada correspondentes ao preditor, onde os *biases* e pesos não devem ser alterados, e atribuindo-lhe um valor não nulo para as camadas correspondentes ao controlador (esse fato pode ser realizado conforme a Equação A1.4). Ao final do treinamento, a parcela correspondente ao controlador é desacoplada da rede concatenada, possibilitando-lhe o seu uso em simulações ou em operações reais.

Por fim, os dados para treinar o sistema, constituído por controlador (direto ou indireto) e preditor, podem ser obtidos tendo a identidade como referência, no caso de se buscar uma inversa à esquerda, ou pode-se utilizar outro modelo de referência quando se quiser associar à rede concatenada uma dinâmica particular.

6.3.2 Controlador Direto

Uma abordagem diferente no que se refere ao projeto do controlador é especificada a seguir. Essa forma de obtenção do controlador independe do preditor anteriormente treinado,

não sendo necessário criar para isso nenhuma rede de concatenação. Logo, somente uma rede é criada: aquela representante do controlador neural.

Como no caso da seção anterior, a quantidade de neurônios e o número de camadas do controlador seguem o mesmo processo heurístico de definição empregado no projeto de qualquer rede; esses atributos, assim como os parâmetros de treinamento, podem ser diferentes ou iguais aos do preditor implementado.

Objetivando-se conseguir um controle livre de erros advindos de aproximações neurais, o controlador passou a ser treinado diretamente a partir dos dados coletados e dos gerados a partir do modelo das condições singulares de giro, conforme mapeamento representado na Equação 6.1. Assim, fica a cargo de trabalhos futuros o desenvolvimento de controladores advindos de modelos neurais projetados com variados horizontes de predição.

Os resultados do treinamento e da validação desse tipo de controlador para a aplicação do presente trabalho, usando o horizonte de predição escolhido para o preditor de melhor desempenho, são mostrados nas próximas seções.

6.4 Treinamento do Controlador

O controlador neural direto foi sintetizado, de forma empírica, com a seguinte topologia:

- Referência do sistema: $\delta\theta_2^*(k+h)$;
- Número de camadas ocultas: 2;
- Quantidade de neurônios por camada: 7, 11 e 1, respectivamente;
- Função de criação da rede multicamada *feedforward*: *newff*;
- Função de ativação por camada oculta: sigmoidal bipolar – *tansig*;
- Função de ativação na camada de saída: linear – *purelin*;
- Algoritmo de treinamento: *Backpropagation* com taxa de aprendizado adaptativa e adição de momento;
- Função de treinamento: *traingdx* / Função de aprendizado: *learngdm*;
- Modo de treinamento: *batch*;
- Funções de pré-processamento: *fixunknowns, mapminmax, removeconstantrows*;
- Funções de pós-processamento: *removeconstantrows, mapminmax*;
- Função de *performance*: *mse*;
- Sem Parada Antecipada (*Early Stopping*): mesma justificativa no caso anterior do preditor.
- Função de inicialização aleatória dos pesos e *biases* da rede.

Vale destacar que as mesmas considerações, referentes ao estabelecimento do número de épocas e do erro (*mse*) desejado, destinadas ao preditor são válidas quando do treinamento do controlador.

Caso esse controlador fosse utilizado juntamente com o preditor anteriormente projetado, a rede concatenada apresentaria a seguinte arquitetura neural:

- Rede do tipo estática *feedforward*;
- 4 entradas: $\theta_1(k)$, $\theta_2(k)$, $\delta\theta_1(k)$, $\delta\theta_2(k)$ mais o sinal de referência: $\delta\theta_2^*(k+h)$;
- 5 camadas ocultas;
- 2 saídas: $\delta\theta_1(k+h)$ e $\delta\theta_2(k+h)$;
- Quantidade de neurônios, respectivamente por camada:
  - 7, 11 e 1, referentes ao controlador;
  - 7, 11 e 2, referentes ao preditor.

A Figura 6.1 mostra a parte da *Interface* implementada destinada a criar, a parametrizar, a treinar e a testar a generalidade de um controlador de horizonte H.

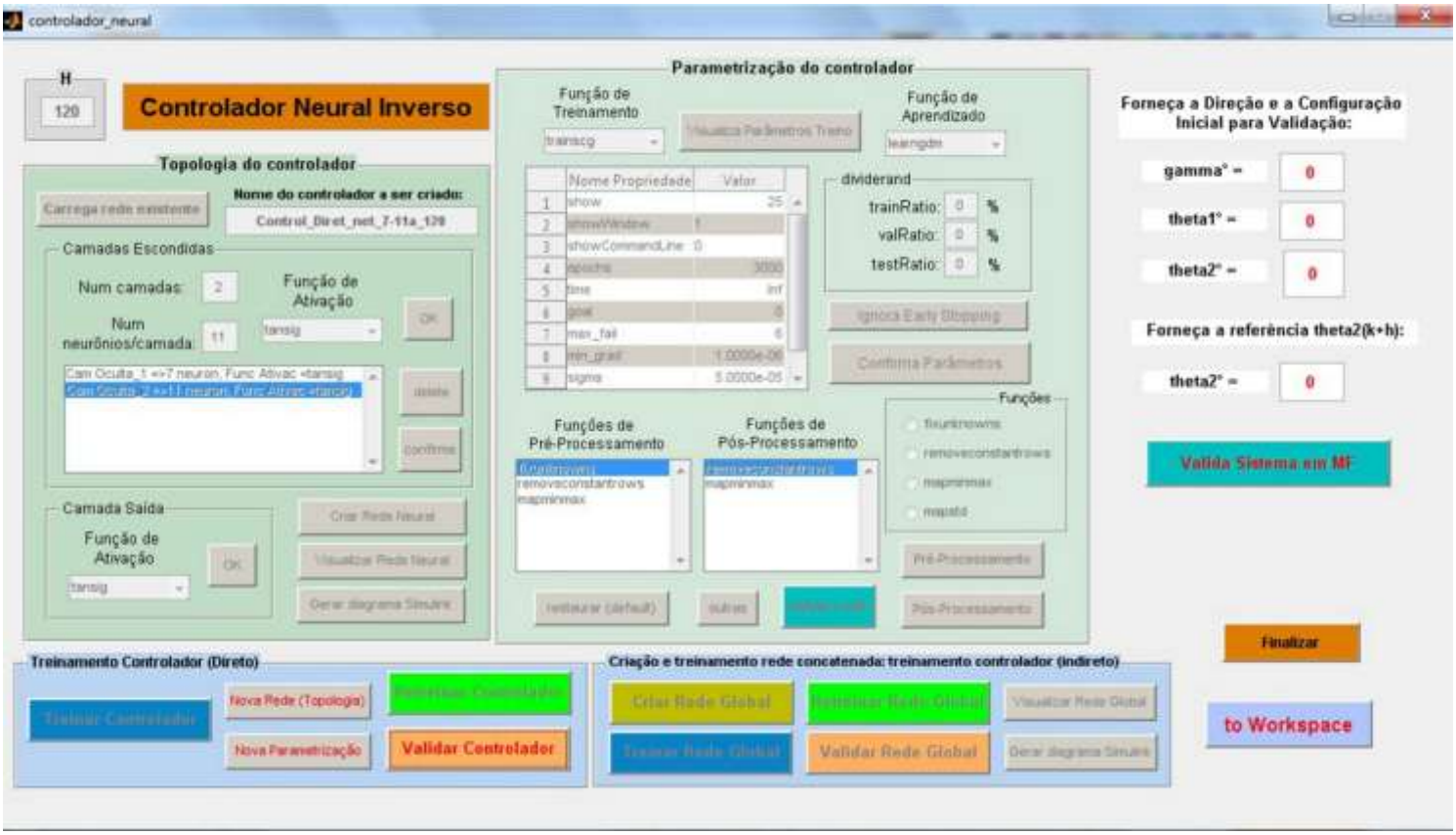


Figura 6.1 Ilustração da *Interface* implementada destinada a criar, a parametrizar, a treinar e a testar a generalidade de um controlador de horizonte H *(controlador.m, .fig)*.

Na Figura 6.2, são apresentados os resultados relativos ao Teste de Generalidade do controlador treinado de horizonte de 120*ms*, correspondentes aos erros e às comparações entre a saída obtida pelo controlador e a saída desejada. Nos gráficos, os resultados em graus (eixo vertical) são mostrados para cada padrão (eixo horizontal).

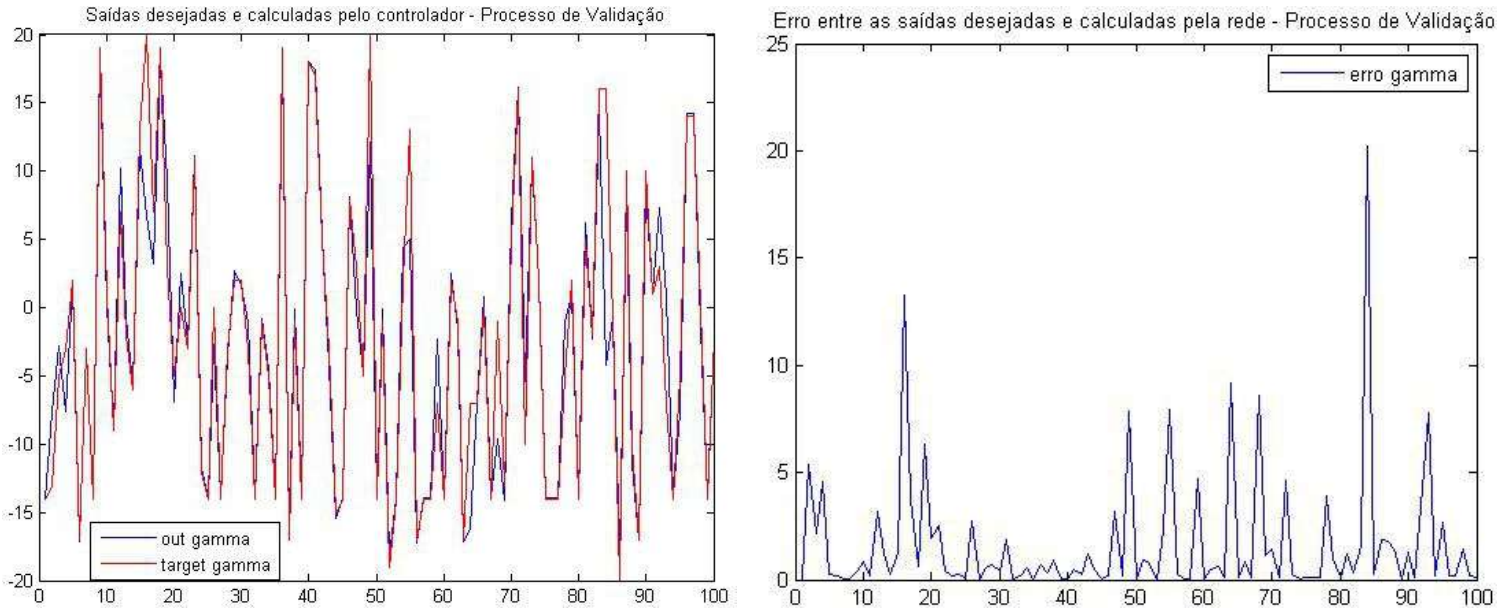


Figura 6.2 Resultados do Teste de Generalidade para o controlador: comparação entre a saída, $\gamma(k)$, calculada e a desejada (à esquerda); e erro absoluto entre os valores obtidos e objetivados da saída (à direita) – exibição dos 100 primeiros valores.

6.5 Validação do Controlador

A fase de validação do controlador transcorreu em simulação, em malha fechada, fazendo uso inicialmente do preditor neural implementado. Posteriormente, o preditor analítico, sintetizado conforme equacionamento do Capítulo 4 (Seção 4.1), foi incorporado na malha junto ao controlador neural e os resultados foram analisados.

O objetivo dessa fase seria o de simular o controle de um RMMA efetuando movimentos à ré a partir de configurações convexas, de modo a manter o giro ou realizar adequadamente a transição entre giros (mantendo a convexidade) e a evitar situações de configurações críticas e de *jackknife*.

A Figura 6.3 ilustra um exemplo de um movimento de RMMA na ausência de controle. Percebe-se a ocorrência de uma situação crítica e de *jackknife*, o que é facilitada pela própria geometria do RMMA. Fica evidente que, durante o movimento de translação à ré, a

rotação de um elemento da cadeia em um sentido força a rotação do elemento subsequente no sentido contrário, o que dificulta a ação de controle.

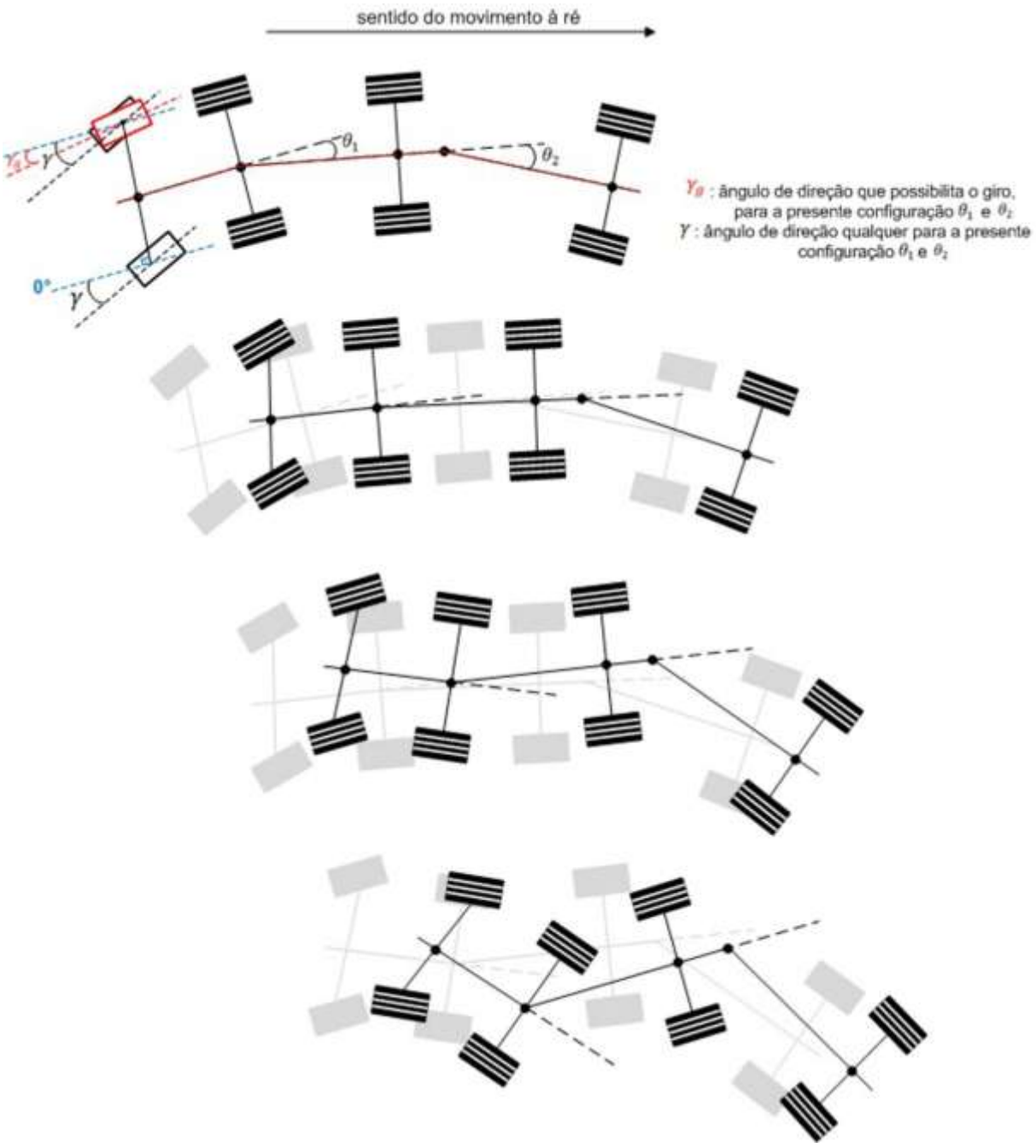


Figura 6.3 Exemplo de movimento de RMMA na ausência de controle.

Para a validação do controlador, a *Interface* possibilitava ao usuário fornecer as configurações iniciais convexas, juntamente com a referência do sistema. O esquema da Figura 6.4 caracteriza o processo de validação do controlador em malha fechada.

Na Figura 6.4, mesmo não sendo representados na malha, foram implementados saturadores nas entradas do controlador, ficando essas limitadas aos seus correspondentes valores máximos e mínimos encontrados na base de treinamento.

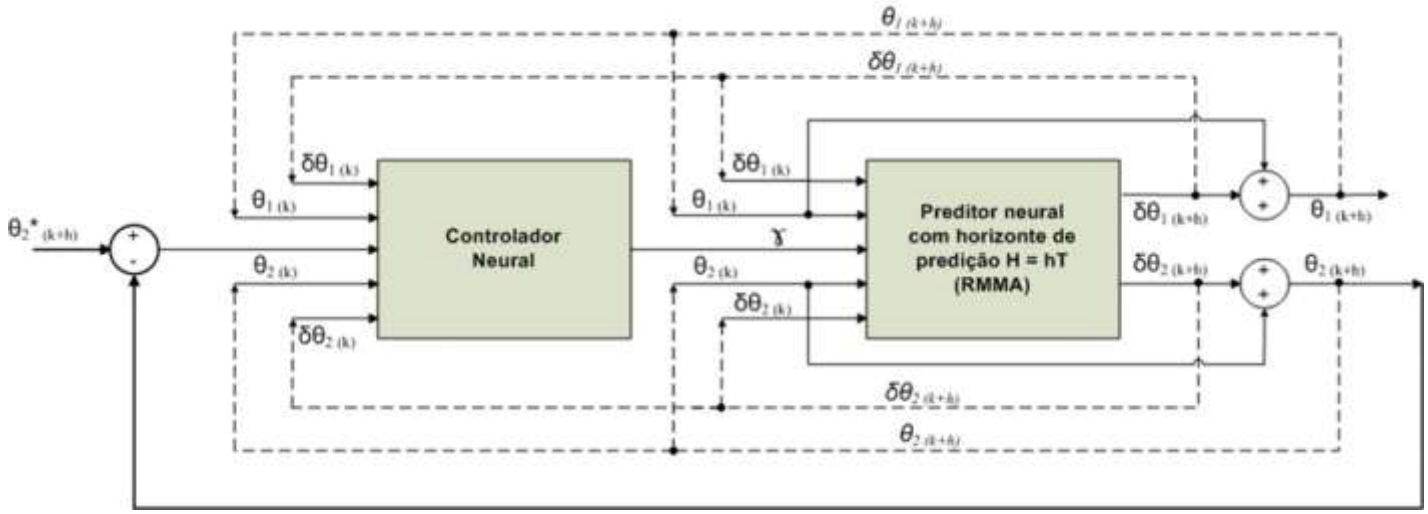


Figura 6.4 Esquema de validação, em malha fechada, do controlador.

A velocidade do modelo analítico foi estimada experimentalmente fazendo-se uso do próprio protótipo. Para uma distância pré-fixada, mediu-se o tempo gasto pelo RMMA até se completar o percurso. Esse procedimento foi repetido algumas vezes e, por fim, o tempo considerado correspondeu à média daqueles medidos. A velocidade obtida foi, então, de $0.1667 m/s$. No esquema representado acima, ao incorporar o preditor analítico no lugar do neural, acrescenta-se uma entrada adicional ao modelo, correspondendo à velocidade estimada.

Na sequência, são mostrados os resultados alcançados na validação do controlador, para diferentes configurações iniciais e referências impostas, fazendo-se uso dos preditores neural e analítico. Nos gráficos, os valores em azul claro representam a resposta do controlador, ou seja, correspondem aos valores em graus do ângulo de direção, γ, do RMMA; os valores em vermelho (na forma de "x") correspondem à referência, $\theta_2{}^*$, imposta ao sistema na sua forma angular em graus (e não em variação); as demais curvas representam as respostas do sistema para certa condição inicial $(\gamma_{inicial}, \theta_{1_{inicial}}, \theta_{2_{inicial}})$ e uma dada referência $(\theta_2{}^*)$: os valores em verde correspondem ao ângulo θ_1, em graus, e os em azul escuro, ao θ_2, em graus. Para cada trajetória, o tempo decorrido é calculado multiplicando-se o número da iteração corrente (equivalente ao número do padrão atual – representado no eixo horizontal) por $120 ms$, o qual corresponde ao horizonte escolhido. Vale lembrar que os

valores críticos dos ângulos de configuração, para tração traseira, são $\theta_{1c} = \pm 32.0257°$ e $\theta_{2c} = \pm 54.8926°$.

Preditor neural com horizonte de predição de 120*ms*:

Teste 1: condição inicial ($\gamma = +15°, \theta_1 = -20°, \theta_2 = -39.7°$) e referência ($\theta_2{}^* = -41°$):

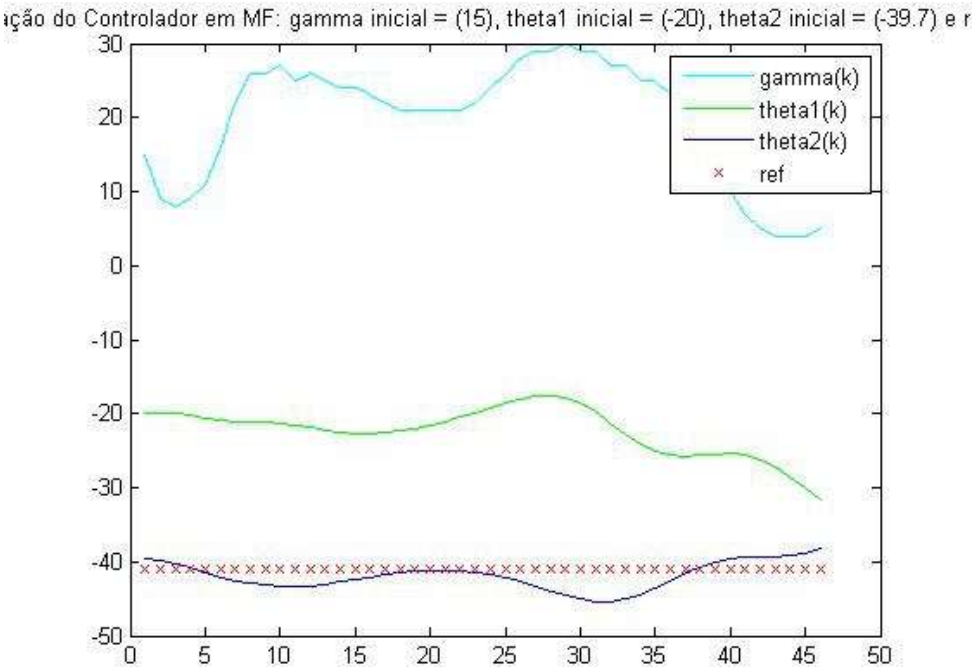


Figura 6.5 Resultados da simulação 1.

Teste 2: condição inicial ($\gamma = +16°, \theta_1 = -24°, \theta_2 = -43.4°$) e referência ($\theta_2{}^* = -44°$):

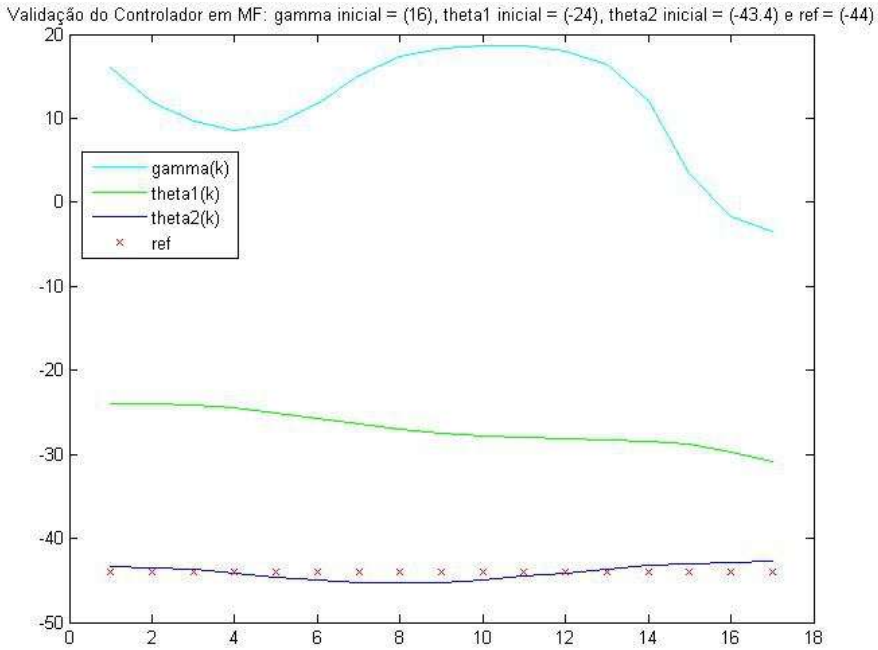


Figura 6.6 Resultados da simulação 2.

Teste 3: condição inicial ($\gamma = +16°, \theta_1 = -25.5°, \theta_2 = -43.4°$) e referência ($\theta_2{}^* = -43.4°$):

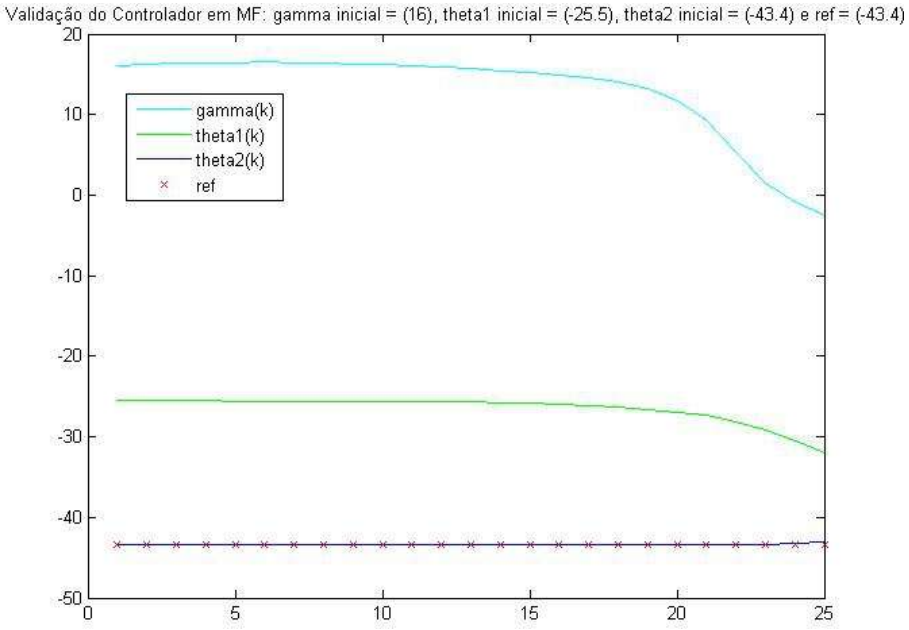


Figura 6.7 Resultados da simulação 3.

Teste 4: condição inicial ($\gamma = +4°, \theta_1 = -7°, \theta_2 = -10.5°$) e referência ($\theta_2{}^* = -10.5°$):

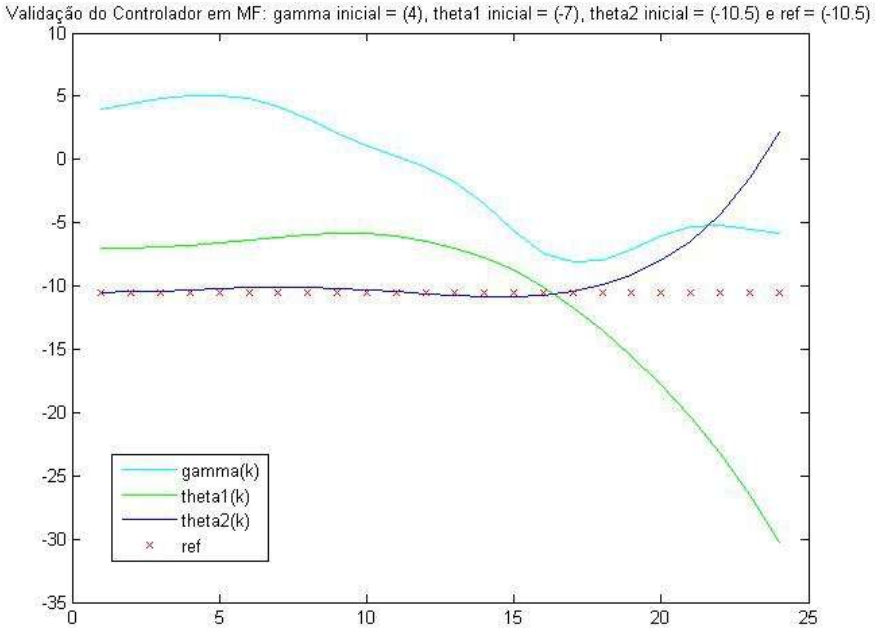


Figura 6.8 Resultados da simulação 4.

Teste 5: condição inicial ($\gamma = -5.5°, \theta_1 = -6.5°, \theta_2 = -20°$) e referência ($\theta_2{}^* = -22°$):

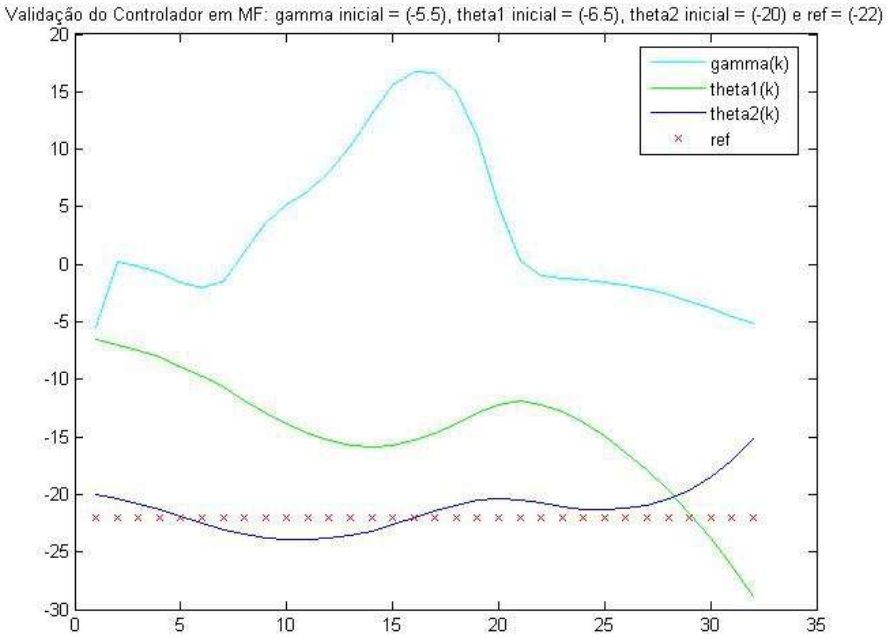


Figura 6.9 Resultados da simulação 5.

Teste 6: condição inicial ($\gamma = -5°, \theta_1 = -6.5°, \theta_2 = -20°$) e referência ($\theta_2{}^* = -22°$):

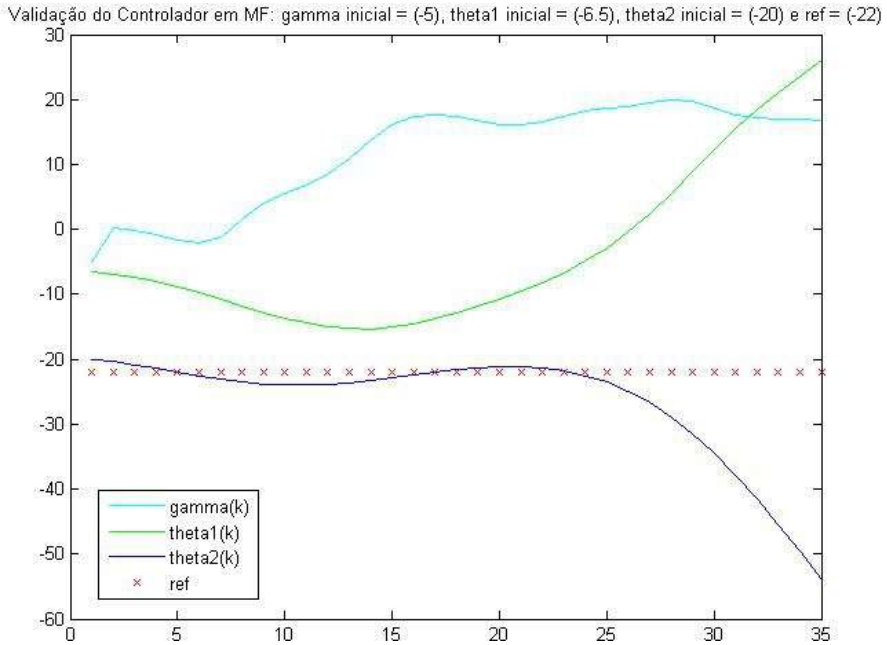


Figura 6.10 Resultados da simulação 6.

Preditor Analítico com horizonte de predição de 120*ms*:

Teste 7: condição inicial ($\gamma = +15°, \theta_1 = -20°, \theta_2 = -39.7°$) e referência ($\theta_2{}^* = -41°$) – *ver Teste 1*.

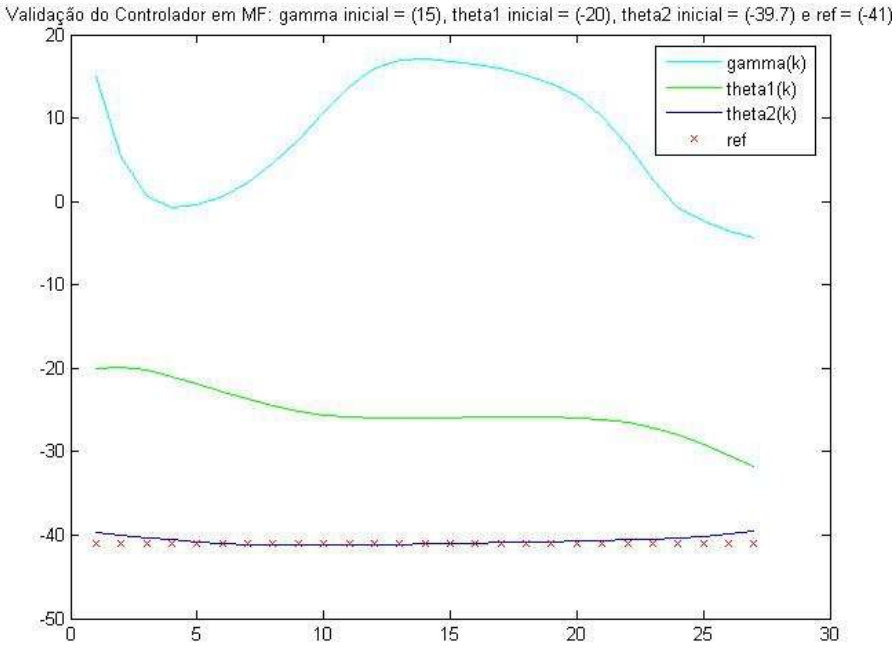


Figura 6.11 Resultados da simulação 7.

Teste 8: condição inicial ($\gamma = +16°, \theta_1 = -24°, \theta_2 = -43.4°$) e referência ($\theta_2{}^* = -44°$) – *ver Teste 2*:

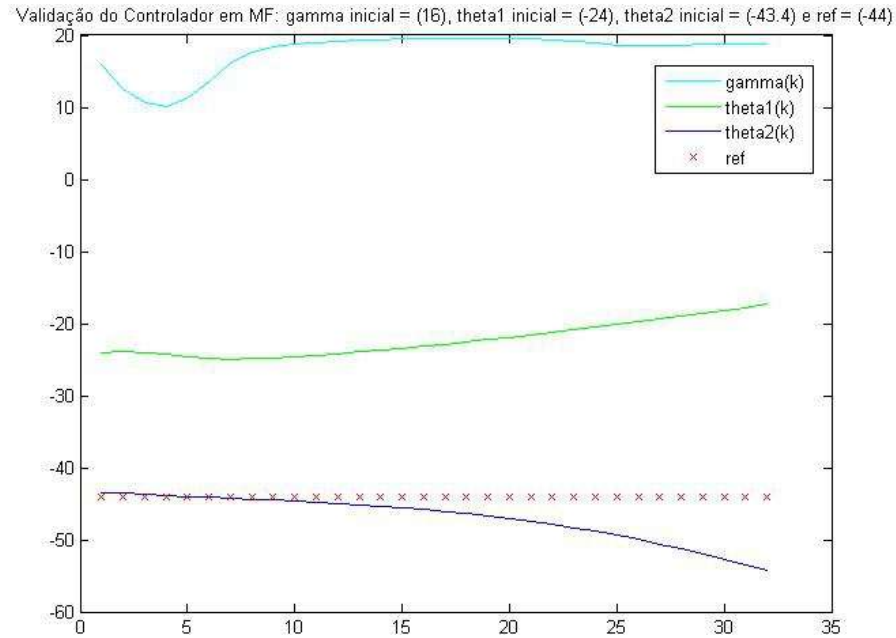


Figura 6.12 Resultados da simulação 8.

Teste 9: condição inicial ($\gamma = 12°, \theta_1 = -18.86°, \theta_2 = -30°$) e referência ($\theta_2{}^* = -30°$):

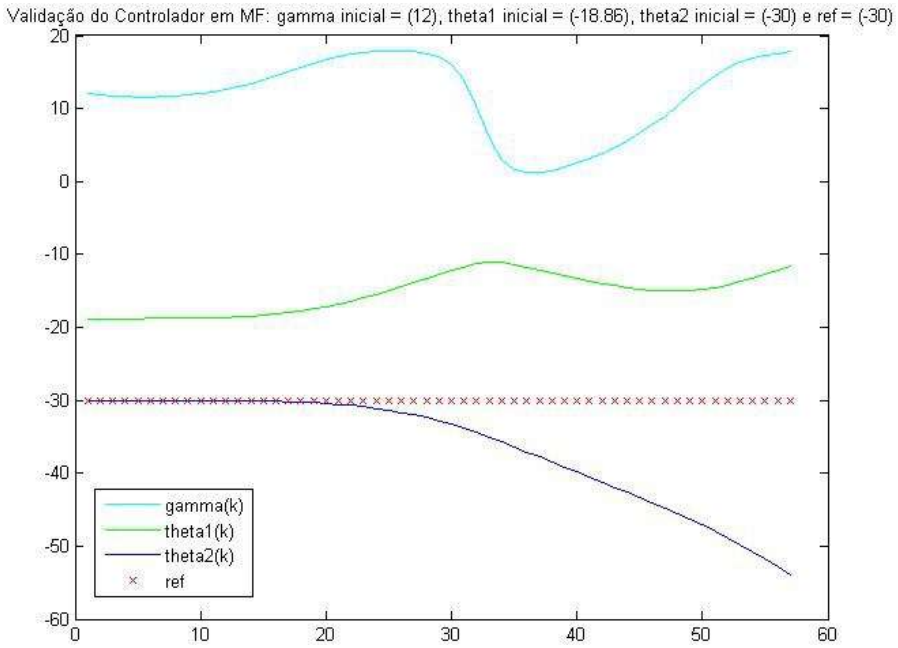


Figura 6.13 Resultados da simulação 9.

Após análise das respostas geradas, a partir da realização dos testes de simulação em malha fechada usando ambos os tipos de preditores, verifica-se que o controlador possui um comportamento satisfatório, pois consegue manter o ângulo de configuração, θ_2, próximo ou equivalente à referência imposta ao sistema, durante intervalos de tempo razoáveis, iguais ou maiores ao horizonte de simulação do preditor, mesmo apesar de cada simulação ser paralisada em virtude de um dos ângulos de configuração atingir seu valor crítico correspondente ocasionando o *jackknife*.

Por outro lado, em experimentos práticos com o protótipo, a ação inadequada de controle foi proveniente principalmente da utilização de uma base de dados que, para o controlador, se revelou insuficiente e não totalmente representativa (diferentemente do ocorrido com os preditores, para os quais o mesmo conjunto de dados se mostrou o oposto), e não causada em virtude da estratégia de controle adotada ou da arquitetura neural projetada do controlador. Acredita-se, por experiência adquirida e por meio de experimentos realizados, que modificações na estrutura neural e na parametrização de treinamento do controlador sintetizado não acarretariam na solução do problema encontrado para o caso prático, apesar de tal fato poder prover melhorias em algumas situações testadas (ou não). Algumas justificativas são levantadas, a seguir, com base nas características observadas da massa de dados.

O controlador, por necessitar de variações angulares como entrada, fica sujeito a trabalhar com valores de variações, os quais, mesmo com a modificação do tempo de amostragem ou do horizonte de predição, podem ser ainda considerados pequenos. Assim, qualquer variação angular fora dos limites de domínio a partir do qual o controlador foi treinado pode não garantir a interpolação e reconhecimento de tal entrada, de modo a não permitir ser tomada a atitude de controle correta, ou seja, não sendo retornado o ângulo de direção do robô, γ, correto.

Além disso, diante de uma base de dados insuficiente e não totalmente representativa para o controlador em experimentos reais com o protótipo, algumas configurações e ações de controle podem não ter sido coletadas para o treinamento do controlador. Assim, um maior fracionamento dos domínios dos ângulos de configuração e do ângulo de direção acarretaria em uma maior quantidade de condições iniciais de movimentos e poderia garantir uma maior representatividade da base. Adicionalmente, seria aconselhável a execução de movimentos mais longos, variando-se o ângulo da direção ao longo dos mesmos. Logo, como alternativa, fica a cargo de novos trabalhos o planejamento e a formalização de outra metodologia de coleta de dados para o projeto de controladores neurais, visando à otimização da base de dados. Para isso, dispositivos (tais como *joysticks*) que permitam uma aquisição de dados mais adequada e mais abrangente podem ser usados. Por fim, testes mais detalhados com outras topologias, parametrizações e horizontes de predição para os controladores não deixam de ser importantes e podem ajudar a melhorar o desempenho do sistema.

Além da massa de dados, outros fatores podem influenciar o resultado de controle em experimentos com o robô em malha fechada. Neste trabalho, tem-se uma aplicação real em um protótipo multiarticulado com forte restrição no ângulo de controle. Verifica-se, de fato, que essa restrição simplifica a obtenção de preditores, mas complica muito a síntese de controladores, pois aumenta as restrições de configurações críticas. Logo, essa restrição poderia ser relaxada por modificação de *hardware* mecânico, o que possibilitaria maior manobrabilidade do robô.

Outro aspecto a se considerar é que, devido a fatores físicos, inerentes ao próprio protótipo utilizado (tais como folgas mecânicas dos potenciômetros, mau contato e isolamento incorreto de fios), ruídos podem ser inseridos pelo sensoriamento empregado na medição das variáveis, agravando o problema. Logo, no caso mais geral, são propostas novamente melhorias e modificações no *hardware* do protótipo utilizado, a fim de se evitar constantes correções manuais antes da realização de experimentos.

Como proposta adicional, um sistema de controle empregando modelo de referência com cancelamento de ruídos e distúrbios poderia ser futuramente esquematizado, conforme proposto por Widrow (1996) e apresentado na Figura 6.14.

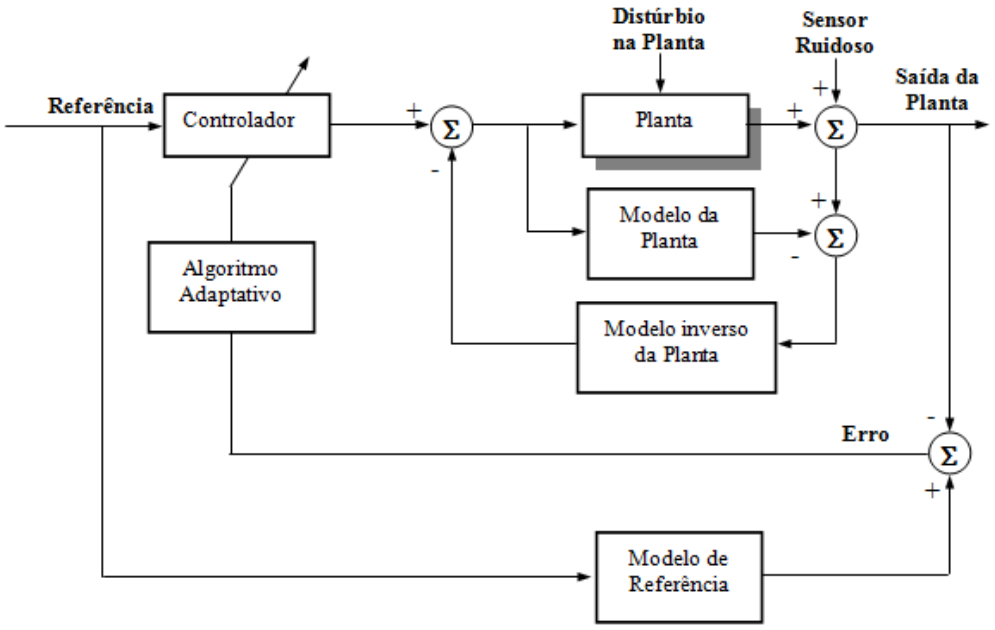


Figura 6.14 Esquema de um sistema de controle inverso via modelo de referência com cancelamento de distúrbio e ruído.

Objetivando-se prover melhorias no controle, sintonizadores PID podem ser sintonizados, via algoritmo genético, por exemplo, e, posteriormente, incorporados ao sistema linearizado, formado pela junção do controlador mais preditor, fechando-se a malha resultante e fazendo com que a variação angular, $\delta\theta_2$, tenda a se estabilizar em direção à referência. Esse fato será tema de trabalhos futuros.

Finalizando, a simples constatação da facilidade adicionada pelo controle, na realização de manobras virtualmente impraticáveis a um condutor inexperiente, contudo elementares à navegação de robôs multiarticulados efetuando movimentos à ré, reforça as expectativas iniciais da contribuição deste trabalho de evoluções na automação da navegação desses veículos. Novamente, espera-se que resultados como este sejam desdobrados em melhorias no transporte de cargas tanto na cadeia logística rodoviária quanto em células flexíveis de manufatura.

Capítulo 7: Conclusões

Ao término do processo de pesquisa, análise e síntese que compõe o estudo descrito ao longo deste trabalho, faz-se necessária a retomada de alguns dos objetivos específicos e gerais perseguidos ao longo da confecção do mesmo, de modo a sintetizar os sucessos alcançados, as barreiras transpostas, os imprevistos observados e as lacunas a explorar.

A ideia empregada neste trabalho originou-se basicamente da visualização de uma demanda por melhorias na área da navegação robótica especificamente direcionada aos veículos ou robôs móveis multiarticulados. Tais melhorias visam fornecer meios que permitam à cadeia logística rodoviária e às atividades manufatureiras adequar seus sistemas de transporte de modo a reduzir custos, a otimizar processos e a elevar a eficiência dos mesmos.

O presente trabalho iniciou sua fundamentação teórica com base na pesquisa e análise de um protótipo de RMMA e de sua cadeia cinemática, tendo sido fundamental conhecer a natureza do mesmo e identificar, além de seus componentes e características, suas variáveis de configuração, os quais constituem integrantes essenciais nos processos de modelagem e controle neural.

O protótipo utilizado apresentou desempenho condizente com as finalidades a que era destinado. Sendo o mesmo destinado ao estudo da dinâmica de RMMA's de grande similaridade a veículos multiarticulados reais, a partir do protótipo foi possível extrair o conhecimento empregado na modelagem do sistema e, posteriormente, na síntese e na validação do controlador.

Um papel de extrema importância do protótipo utilizado consistiu em permitir a geração de informações necessárias à confecção das bases de dados a serem utilizadas. Na ausência de uma metodologia consolidada na literatura para a coleta dos dados, foi empregada, no trabalho, uma estratégia para a elaboração da base de treinamento dos modelos. Essa consistia no fracionamento do universo de possibilidades dos ângulos de configuração e da direção em partições definidas conforme o conhecimento do especialista, de modo a retratar a dinâmica do sistema. Essas partições eram, então, combinadas a fim de gerar condições iniciais de movimentos do RMMA. Acreditava-se, portanto, no fato de essa

base ser suficiente em tamanho e suficiente para gerar, aos modelos, informações representativas de movimentos convexos possíveis.

Em problemas envolvendo sistemas complexos e não-lineares, de difícil modelagem ou abordagem analítica, as soluções denominadas inteligentes têm se tornadas cada vez mais atrativas. Nesse contexto, fatores inerentes ao sistema real estariam representados no modelo. O presente trabalho buscou empregar as redes neurais na busca de soluções inéditas e de resultados conclusivos relacionados ao problema de modelagem e controle à ré de um RMMA com três graus de liberdade.

Todo o embasamento teórico, encontrado no Capítulo 3 (e também no Apêndice 1), foi fundamental basicamente para tornar mais transparente ao usuário alguns fatores antes implícitos e abstratos tais como: a identificação dos componentes das redes neurais e como os mesmos se relacionam e se comunicam; o funcionamento dos algoritmos usados para o aprendizado neural e como tais métodos percorrem e modificam as estruturas das redes; entre outros; além de tal embasamento ter sido essencial a um melhor entendimento acerca da metodologia empregada para o projeto de modelagem e controle neural, focos do presente trabalho.

Este trabalho apresentou uma ferramenta original desenvolvida para facilitar o usuário nas exaustivas tarefas de coleta e tratamento de dados, treinamento e validação de redes neurais que poderiam atuar como preditores ou controladores completos, a horizonte fixo, no espaço de configuração de RMMA's. Foi apresentado também um modelo genérico original para as singularidades, utilizado para prover dados de giro (cuja obtenção a partir de sistemas reais em malha aberta se mostrou inviável) para um treinamento mais abrangente. Este mesmo modelo pôde ser utilizado para a obtenção dos ângulos críticos, ou condições de contorno, utilizados na síntese de controladores para um dado robô. Os detalhes de implementação da *Interface*, a dedução detalhada dos modelos das condições singulares e dos ângulos críticos e os ensaios práticos para a avaliação de desempenho das ferramentas propostas foram objetos de apresentação deste trabalho em particular.

No que diz respeito à modelagem do sistema, os preditores neurais completos a horizontes fixos projetados apresentaram resultados satisfatórios, confirmando, desta maneira, a eficiência da proposta estabelecida. Os novos modelos propostos para as condições singulares e para os ângulos críticos foram validados em uma situação real. A estratégia adotada para a coleta de dados foi suficiente para garantir o aprendizado da rede, durante seu treinamento, e sua posterior capacidade de generalização.

As estratégias adotadas e os resultados gerados durante a fase de validação dos preditores comprovaram a adequação da base de dados para tal feito. A metodologia empregada baseada na reprodução de algumas trajetórias possibilitou a escolha de um preditor com um horizonte de predição definido e com desempenho bastante satisfatório dentro de certo intervalo de tempo (considerando aplicações em simulação), onde o erro acumulado não promoveria perturbações consideráveis nos valores dos ângulos de configuração, $\theta_1(k+h)$ e $\theta_2(k+h)$, gerados pela rede treinada.

A justificativa para o emprego de controladores neurais, ao invés de soluções clássicas como controladores do tipo PID, encontra-se na constatação empírica de que, para plantas de controle como a tratada neste trabalho, onde o sistema é não-linear e o modelamento é muito complexo, o desempenho obtido com o emprego de controladores clássicos é, em geral, inferior (quando possível de implementar) se comparado ao de controladores inteligentes.

Dentre as várias implementações de controle que fazem uso das redes neurais, optou-se pelo emprego do controle neural inverso, a horizonte fixo, no espaço de configuração. Em linhas gerais, para o projeto do controlador, foram utilizados, durante o treinamento, dados provenientes diretamente do protótipo e dos modelos das condições singulares de giro, livres, portanto, de erros provenientes do preditor sintetizado.

Tendo por base o controlador originalmente projetado, seu desempenho foi medido diante da metodologia empregada durante a sua validação. Os experimentos transcorreram em simulação com a malha de controle fechada e, após análise dos mesmos, verificou-se que os resultados obtidos foram adequados e satisfatórios. Observou-se para todos os casos testados que, ao longo das trajetórias, interpretações e reações corretas foram tomadas pelo controlador além do intervalo de tempo esperado (definido pelo horizonte de validade do preditor). Já no caso prático, a falta de manutenção de ações de controle corretas durante todo o percurso resultou em uma situação de *jackknife*.

7.1 Trabalhos Futuros

Uma das metas deste projeto consistia em tirar proveito do protótipo já implementado, a fim de gerar subsídios o suficiente para gerar o máximo de resultados inéditos e satisfatórios. Assim, o presente trabalho partiu de estratégias simplificadas visando possibilitar a geração de resultados conclusivos para a aplicação abordada, durante todas as

suas etapas, e visando validar o uso da ferramenta criada a servir de base e auxílio a outros trabalhos correlacionados.

A proposta deste trabalho passou pelo conhecimento de uma arquitetura subdividida em níveis de supervisão e controle, cabendo ao primeiro a implementação de regras de navegação e tomada de decisões, e ao segundo a operacionalização das diretrizes traçadas e supervisionadas. Porém, o projeto corrente destinou-se somente a explorar o nível de controle, mais especificamente de controle de configuração, deixando como sugestão para trabalhos futuros a implementação do nível de supervisão como forma de orientação de manobras do veículo em tarefas como estacionamento ou de navegação em ambientes estruturados com obstáculos estáticos ou dinâmicos.

Considerando o problema real de controle, seria aconselhável buscar alternativas a fim de trazer, se possível, melhores resultados ou de tentar ajudar a explicar alguns insucessos ocorridos, gerando, se for o caso, novas propostas e novas estratégias a serem utilizadas em trabalhos futuros. Apesar de se apresentar, neste trabalho, os procedimentos a serem adotados no projeto de controladores indiretos, obtidos a partir de preditores neurais projetados, cabem a trabalhos futuros a implementação e análise dos mesmos. Além disso, pretende-se futuramente sistematizar a síntese de controladores lineares que garantam robustez em malha fechada do sistema: essa estratégia consiste na introdução de sintonizadores Proporcional-Integrador-Derivativo (PID) sobre o sistema linearizado resultante, composto pelo controlador mais preditor, como forma de minimizar o erro entre o ângulo da última articulação e a referência imposta.

Em termos práticos, quando se utiliza o controlador em um computador que se comunica com o protótipo, existe um atraso, entre a resposta da planta e a correspondente ação de controle para a configuração angular em questão, que ocorre devido ao processamento remoto e à comunicação entre os dispositivos, o que faz com que o problema de controle se agrave. Assim, cabe a trabalhos futuros, também, a implementação de controladores neurais em sistema embarcado no próprio robô, o que permitiria o processamento local, possibilitando, desse modo, a diminuição do tempo de resposta do controlador quando a ele fornecido certa configuração.

Além disso, cabe a trabalhos futuros fazer uso mais abrangente da *Interface* implementada no projeto de preditores e controladores neurais com diferentes características (topologias, parametrizações, horizontes de predição,...) a serem testados para as mais diversas condições angulares iniciais de movimento e referências. Outras sistemáticas mais detalhadas para a determinação da melhor estrutura (topologia e parametrização) de preditor,

para um dado robô, bem como para o estabelecimento de controladores neurais adequados poderão também ser exploradas em trabalhos futuros.

Por fim, acredita-se que esse trabalho possa ser uma base satisfatória na busca de soluções em diversas aplicações futuras de robôs móveis multiarticulados. Vale ressaltar que os resultados alcançados são inéditos e relativamente satisfatórios, considerando o pequeno tamanho da base de dados coletada e a baixa representatividade dos dados nela contidos para o caso de controle do protótipo de RMMA, pois a síntese de redes neurais estáticas na aproximação de preditores e controladores completos, nas condições aqui estabelecidas, ainda não foi abordada na literatura. Apesar de o problema de controle não ter sido solucionado por completo (como se percebe no caso prático), o trabalho possibilitou o conhecimento de metodologias e abordagens para tratamento do mesmo, esperando-se, com isso, fomentar novas pesquisas na área, bem como gerar uma fonte de referência e de subsídios quando da realização de trabalhos posteriores.

Referências Bibliográficas

Ballini, R.; Gomide, F. (2001). A Recurrent Neuro-Fuzzy Network Structure and Learning Procedure. *10th IEEE International Conference on Fuzzy Systems – Fuzzy-IEEE*.

Barreto, T. R. (2010). Desenvolvimento de Algoritmos e Ferramentas de Síntese e Validação das Estratégias para o Controle em Manobras ou Navegação de Veículos ou Robôs Móveis Multiarticulados. *Dissertação (Mestrado) – Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal do Espírito Santo, Vitória*.

Bertolani, D. N. (2011). Robôs Móveis Multiarticulados - Proposta de Algoritmo Feedback Não-Linear e Validação Comparativa com Soluções Fuzzy em Protótipo Real. *Monografia (Engenharia de Computação) – Departamento de Informática, Universidade Federal do Espírito Santo, Vitória.*

BgmRodotec. (2011). Associação Nacional do Transporte de Cargas e Logística (NTC & Logística) [Online]. Disponível em: <http://www.portalntc.org.br/>. 12 Agosto 2011.

Demcenko A., Tamosiunaite M., Vidugiriene A., Saudargiene A. (2008). Vehicle's Steering Signal Predictions Using Neural Networks, *IEEE Intelligent Vehicles Symposium*.

Ferreira, E. P.; Bertolani, D. N.; Pandolfi, F. (2011). Equações De Movimento à Ré de Robôs Móveis Multiarticulados, no Espaço de Configurações e no Espaço de Tarefas com Tração Dianteira ou Traseira e Ligações On-Axle ou Off-Axle – Controle Feedforward, *X Congresso Brasileiro de Inteligência Computacional (CBIC), Fortaleza, Ceará.*

Ferreira, E. P.; Kulitz H.; Silva, E. B.; Pinheiro, M. (2002). Modeling and Simulating Backward of Multiarticulated Mobile Robots or Vehicles, *Fifth World Congress on Computational Mechanics, Viena - Austria*.

Ferreira, E. P.; Kulitz, H. R.; Silva, E. B.; Pinheiro, M. (2004). Modeling and Simulating Movements of Multi-Articulated Mobile Robots or Vehicles - Analytical and Fuzzy Approach, *Chapter of the book: Computational Mechanics in Vehicle Systems Dynamics. Ed. Londres: Taylor & Francis Group plc*, v.40, pp. 51–70.

Ferreira, E. P.; Lamego, M. M.; Widrow, B. (1999). Neurointerfaces for Semi-Autonomous Object Moving Systems, *In: Proceedings of the 14th World IFAC Congress. Beijing, China: Elsevier Science Ltd, Oxford, UK. K*, pp. 155–160.

Ferreira, E. P.; Pandolfi, F.; Reinan, T. (2010). Modelagem Fuzzy de Robôs Móveis Multiarticulados em Manobras Complexas Com Atraso: *Uma Nova Sistemática Para a Descrição de Movimentos e Controladores. XVIII Congresso Brasileiro de Automática, Brasil.*

Haykin, S. (1999). Neural Network: A Comprehensive Foundation. *Prentice Hall, Second Edition*.

Jenkins, R. E.; Yuhas, B. P. (1993). A Simplified Neural Network Solution Through Problem Decomposition: The Case of the Truck Backer-Upper, *IEEE Transactions on Neural Networks*, Vol. 4, no 4.

Kinjo, H.; Maeshiro, M.; Uezato, E.; Yamamoto, T. (2006). Adaptive Genetic Algorithm Observer and its Application to a Trailer Truck Control System, *International Joint Conference Digital*.

Kulitz, H. (2003). Modelagem e controle fuzzy de veículos multiarticulados. *Tese (Doutorado em Automação), Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal do Espírito Santo (PPGEE-UFES), Vitória, ES*.

Lee, C. H.; Teng, C. C. (2000). Identification and Control of Dynamic Systems using Neural Networks, *IEEE Trans. on Fuzzy Systems 3493-366.*

Lim, C. T.; Lee C. S. G. (1996). Neural Fuzzy Systems. *Prentice Hall*, pp. 205-217, pp. 224-249.

Miranda, V. M. (2009). Técnicas de Modelagem e Controle Neural Inverso do Movimento à Ré de um Veículo Articulado Complexo, *Monografia (Engenharia de Computação) – Departamento de Informática, Universidade Federal do Espírito Santo, Vitória.*

Miranda, V. M.; Ferreira, E. P. (2011a). Development of Static Neural Networks as Full Predictors or Controllers for Multi-Articulated Mobile Robots in Backward Movements – New Models and Tools, *9th IEEE International Conference on Control & Automation (ICCA), Santiago, Chile.*

Miranda, V. M.; Ferreira, E. P. (2011b). Development of Static Neural Networks as Full Predictors or Controllers for Multi-Articulated Mobile Robots in Backward

Movements – New Models and Tools, *9th IEEE International Conference on Control & Automation (ICCA), Santiago, Chile*.

Miranda, V. M.; Ferreira, E. P. (2011c). Desenvolvimento de Preditores e Controladores Neurais Completos, a Horizonte Fixo, de Robôs Móveis Multiarticulados, em Movimentos à Ré - Interfaces e Modelos Analíticos dos Ângulos Críticos e das Condições Singulares de Giro, *32th Iberian Latin American Congress on Computational Methods in Engineering (CILAMCE 2011), Ouro Preto, Minas Gerais.*

Miranda, V. M.; Ferreira, E. P.; Villela, C. D. (2011a). Modelos de Ângulos Críticos e Singularidades de Giro na Implementação Real de Preditores Neurais: Robô Móvel Truck-Trailer-Trailer, *X Congresso Brasileiro de Inteligência Computacional (CBIC), Fortaleza, Ceará.*

Miranda, V. M.; Ferreira, E. P.; Villela, C. D. (2011b). Implementação de Preditores Neurais Completos, a Horizonte Fixo, para um Protótipo de Robô Móvel Multiarticulado Truck-Trailer-Trailer, em Movimentos à Ré, *X Simpósio Brasileiro de Automação Inteligente (SBAI), São João Del-Rei, Minas Gerais.*

Nguyen; Widrow, B. (1990). Neural Networks for Self-learning Control Systems. *IEEE Control Systems Magazine,* pp. 18–23.

Pandolfi, F. (2011). Modelagem e Controle de Robôs Móveis Multiarticulados no Espaço de Configurações: Soluções Não-lineares e Fuzzy. *Dissertação (Mestrado) – Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal do Espírito Santo, Vitória.*

Pandolfi, F. (2009). Controlador Fuzzy Centralizado dos Movimentos à Ré de um Robô Móvel Articulado com 3 Graus de Liberdade Gerado com Dados Reais, *Monografia (Engenharia de Computação) – Departamento de Informática, Universidade Federal do Espírito Santo, Vitória.*

Petrov, P. (2010). Nonlinear Backward Tracking Control of an Articulated Mobile Robot with Off-Axle Hitching, *ISPRA'10 Proceedings of the 9th WSEAS International Conference on Signal Processing, Robotics and Automation*.

Pinheiro, M. P. L. (2004). Simulação do Movimento de Robôs Móveis Multiarticulados e Obtenção de um Preditor Fuzzy. *130f. Dissertação (Mestrado) - Programa de Pós*

Graduação em Engenharia Mecânica, Universidade Federal do Espírito Santo, Vitória, ES.

Rampinelli, M. (2006). Análise Comparativa de Estratégias Baseadas em Visão para Estimação de Ângulos de Configuração de Robôs Móveis Multiarticulados, *Monografia (Engenharia de Computação) – Departamento de Informática, Universidade Federal do Espírito Santo, Vitória.*

Rampinelli, M. (2008). Navegação de Robôs Móveis ou Veículos MultiArticulados com Realimentação de Informações Visuais. *84f. Dissertação (Mestrado) – Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal do Espírito Santo, Vitória.*

Rojas, R. (1996). Neural Networks: A Systematic Introduction. *Springer*, pp. 3-169.

Shepherd, G. M. (1990). The Synaptic Organization of the Brain. *Oxford University Press*, 3a ed, Nova York, EUA.

Silva, E. B. (2003). Modelagem e Controle Neural Inverso dos Movimentos de um Veículo Articulado Complexo. *Dissertação (Mestrado) – PPGEE-UFES, Vitória, ES.*

Silva, I. N. (2010). Redes Neurais Artificiais para Engenharia e Ciências Aplicadas: Curso Prático. *Artliber Editora*.

Slack, N. (1997). Administração da produção, *Atlas, First Edition*, pp 257-258.

Vianna, G. A. B. (2011). Transporte Rodoviário. NTC [Online]. Disponível em: <http://www.ntc.org.br/elo_fraco.htm>.

Widrow, B.; Walach, E. (1996). Adaptive Inverse Control. *Prentice-Hall.*

Apêndice 1: O Algoritmo de Aprendizado *Backpropagation*

O esforço computacional necessário para achar a combinação mais adequada de pesos (e *biases*) como solução de um problema aumenta substancialmente quando uma maior quantidade de parâmetros e topologias mais complexas são considerados (Rojas, 1996). Este apêndice apresenta e discute um dos métodos de aprendizado mais conhecidos e amplamente utilizado, capaz de lidar com diversos tipos de problema.

A1.1 O Problema do Aprendizado

O *Backpropagation* é um algoritmo de aprendizado supervisionado aplicado a redes multicamadas *feedforward*. Considere uma rede neural de tal tipo consistindo de *n* entradas, *m* unidades de saída e um número arbitrário de camadas escondidas. Dado um conjunto de padrões de treinamento $\{(x_1, t_1), \dots, (x_p, t_p)\}$, compostos por *p* pares ordenados de vetores n- e m-dimensionais, representando entrada e saída desejada respectivamente, o *Backpropagation* realiza basicamente duas fases de fluxo de dados: primeiramente, a entrada $x_i = (x_{i1}, x_{i2}, \dots, x_{in})$, quando propagada pela rede, gera, ao final, uma saída $o_i = (o_{i1}, o_{i2}, \dots, o_{im})$; em seguida, o erro caracterizado pela diferença entre a saída produzida pela rede, o_i, quando a essa apresentada uma entrada, x_i, e a saída desejada do sistema, $t_i = (t_{i1}, t_{i2}, \dots, t_{im})$, é passado da camada de saída para as camadas anteriores, com os pesos de suas conexões sendo alterados, conforme a propagação (*backpropagation*). A função quadrática do erro é definida abaixo, na Equação A1.1, e calculada criando-se uma extensão da rede neural, conforme mostrada na Figura A1.1.

$$E = E_i = \frac{1}{2}\sum_{j=1}^{m}\left(o_{ij} - t_{ij}\right)^2 \qquad \text{(A1.1)}$$

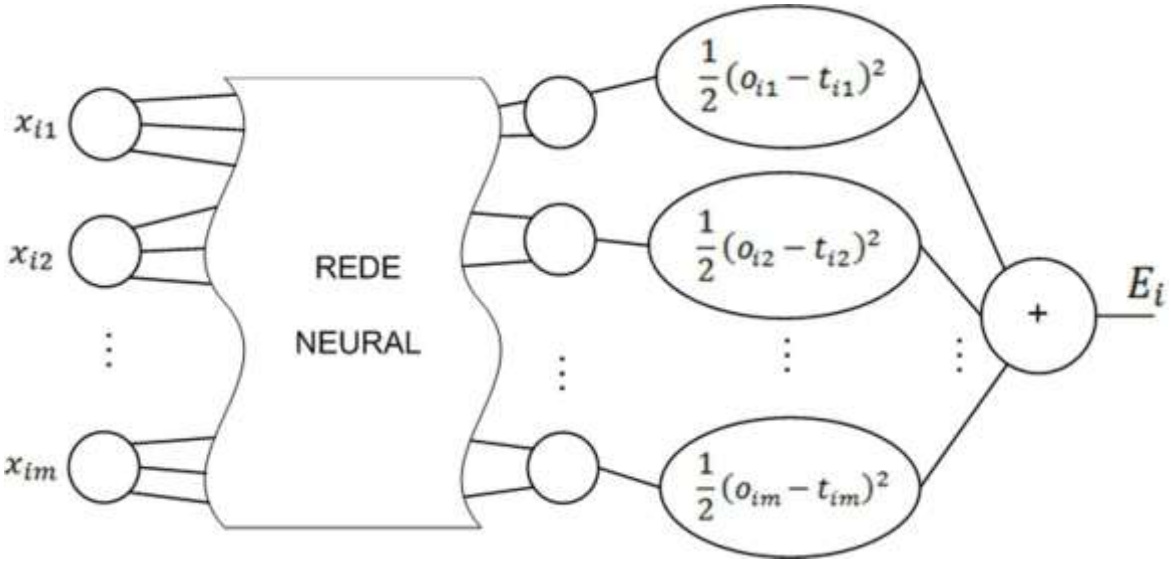


Figura A1.1 Rede extendida para a computação da função de erro.

A rede extendida deve ser criada para cada padrão $(x_i, t_i), i = 1,2, \dots, p$. Assim, o erro total computado, E_T, é formado pela média dos erros gerados para cada par de entrada e saída, conforme a Equação A1.2.

$$E_T = \frac{1}{p}\sum_{i=1}^{p} E_i \tag{A1.2}$$

O *Backpropagation* deve procurar o mínimo da função do erro no espaço de domínio dos pesos usando o método do gradiente descendente. Assim, a combinação de pesos que minimiza a função de erro, ou seja, que idealmente garanta $t_i \approx o_i$, é considerada a solução do problema de aprendizado.

A fim de se minimizar o erro quadrático usando tal método iterativo, é necessário calcular seu gradiente

$$\nabla E = \left(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \dots, \frac{\partial E}{\partial w_l}\right) \tag{A1.3}$$

e atualizar cada peso segundo o incremento

$$\Delta w_i = -\gamma \frac{\partial E}{\partial w_i}, para\ i = 1, \dots, l \tag{A1.4}$$

onde γ representa a taxa de aprendizado (*learning constant ou learning rate*), parâmetro proporcional que define a força do passo de cada iteração na direção negativa do gradiente.

De acordo com a Equação A1.4, quando o valor da taxa de aprendizado é nulo, não ocorre aprendizado da rede, permanecendo inalterado o valor dos pesos; caso contrário, o aprendizado é garantido até que o mesmo se sature, na medida em que valor do peso a ser

incrementado em cada iteração não se modifique, ou, quando da não saturação, o aprendizado é garantido até que o erro desejável seja atingido e, logo, a convergência alcançada.

O aprendizado saturado, representado pela não mais modificação dos pesos e *biases* da rede, quando da posse de uma taxa de aprendizado não nula, não garante a convergência do algoritmo. Essa é influenciada por outros fatores, conforme exemplificados nos capítulos anteriores, objetivando-se um erro mínimo global, entre as saídas desejadas e calculadas pela rede, dentro de um número de épocas adequado pré-estabelecido. Tal convergência não é conseguida quando se atinge um ponto de mínimo local na superfície da função de erro. Por outro lado, não se pode negar que com um conjunto de pesos estacionário, a convergência possa ser alcançada. Nesse caso, essa ocorre quando é alcançado o ponto de mínimo global.

A aproximação executada que estima o gradiente em cada iteração move o vetor de pesos na direção em que a derivada parcial do erro em relação a cada peso é negativa, buscando sempre o ponto de mínimo. Ou seja, considerando-se o erro como uma superfície matemática, a variação dos pesos ocorre na direção de um vale nessa superfície. Quando o gradiente é zero significa dizer que se atingiu um determinado ponto de mínimo, não necessariamente igual ao mínimo global. Nesse caso, alguns fatores de convergência, os quais buscam evitar mínimos locais, devem ser levados em consideração e serão discutidos mais adiante.

Desde que o método *Backpropagation* requer a computação do gradiente do erro a cada iteração, deve-se garantir a continuidade e diferenciabilidade de sua função. Uma função de ativação diferenciável, como a função sigmoidal unipolar (ou sigmóide)[27], apresentada no Capítulo 3 e definida por:

$$s(f) = s_\lambda(f) = \frac{1}{1 + e^{-\lambda f}} \tag{A1.5}$$

torna a função obtida pela rede diferenciável (assumindo que a função de integração em cada unidade computacional seja apenas a soma de suas entradas) devido ao fato de a rede computar apenas composições desse tipo de funções. A função de erro, portanto, também se torna diferenciável.

[27] Será considerada a função de ativação sigmoidal com valor de $\lambda = 1$. Sua derivada, portanto, nesse caso, é expressa por: $s'(f) = \frac{ds(f)}{df} = \frac{e^{-f}}{(1+e^{-f})^2} = s(f).(1 - s(f))$

A1.2 Passos do Algoritmo *Backpropagation*

Antes de se definir o algoritmo, propriamente dito, do método de aprendizado *Backpropagation*, algumas modificações no que se refere à estrutura dos elementos nodais da rede serão evidenciadas, conforme Rojas (1996). Assim, cada nó consistirá de um lado direito, o qual computa a sua função primitiva associada; e de um lado esquerdo, o qual armazena a derivada dessa primitiva para a mesma entrada[28]. Essa representação é chamada diagrama-B e é mostrada na Figura A1.2.

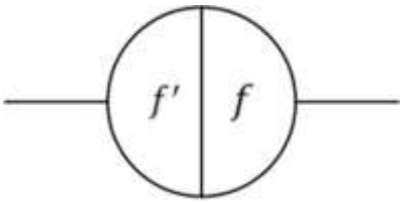


Figura A1.2 Diagrama-B: os dois lados de uma unidade computacional.

Note que a função de integração do nó acima pode ser separada de sua função de ativação dividindo-se o mesmo em dois sub-nós, conforme Figura A1.3, de modo que o primeiro faz a soma das entradas recebidas e o segundo aplica sobre tal resultado a função de ativação, armazenando ainda sua derivada, representada por *s'*. Essa representação, para cada unidade, será implícita de agora em diante.

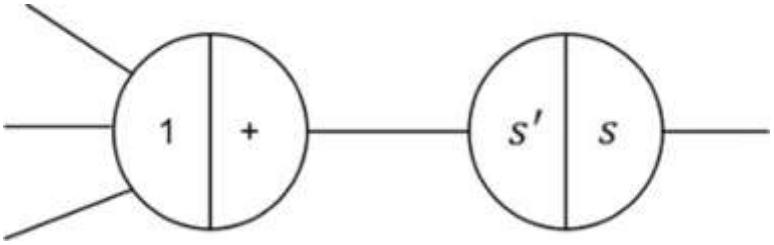


Figura A1.3 Funções de integração e ativação na forma de um diagrama-B.

No primeiro estágio do algoritmo *Backpropagation*, também chamado *feedforward step*, a informação proveniente da entrada nodal é avaliada e processada pela respectiva função primitiva. Essa função é, então, armazenada no lado direito do nó e sua derivada, no lado esquerdo. Apesar de ambos os resultados serem armazenados no nó, apenas o do lado direito é transmitido às outras unidades conectadas. O segundo passo do algoritmo, chamado *backpropagation step*, consiste em percorrer a rede no sentido oposto ao do primeiro passo

[28] No caso da existência de mais de uma entrada por unidade computacional, devemos considerar funções primitivas de várias variáveis armazenadas no lado direito nodal e todas as derivadas parciais dessa função em relação a cada variável, armazenadas no lado esquerdo nodal.

(retropropagação), de modo a usar as derivadas armazenadas. A entrada do lado direito da rede é o valor constante unitário. A informação, agora proveniente do lado direito de cada nó, é multiplicada pelo valor da derivada de sua função primitiva armazenada no seu lado esquerdo. O resultado da multiplicação, também chamado valor traverso (*traversing value*), é transmitido ao próximo nó à esquerda, em direção à entrada da rede. Esses passos são melhor visualizados na Figura A1.4.

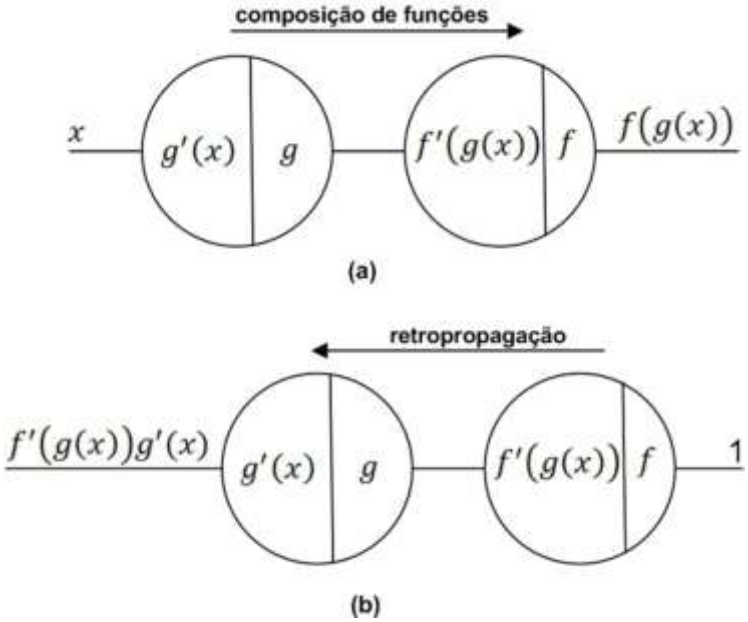


Figura A1.4 Passos do algoritmo *Backpropagation*. (a) *Feedforward step*. (b) *Backpropagation step*.

O *backpropagation step* fornece uma implementação da Regra da Cadeia, explicada no Cálculo Diferencial. Quaisquer sequências de composição de funções podem ser avaliadas para uma determinada entrada e suas derivadas em relação a mesma podem ser obtidas no *backpropagation step.*

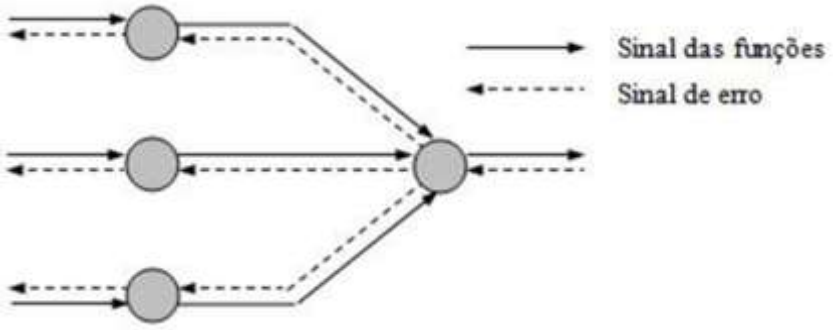


Figura A1.5 Direções básicas dos fluxos de sinais em uma rede multicamadas que faz uso do *Backpropagation*: propagação à frente dos sinais das funções (*forward propagation*) e propagação para trás dos sinais de erro (*backpropagation*).

A1.3 A Implementação do Algoritmo *Backpropagation*

De acordo com Rojas (1996), para se demonstrar o funcionamento do *Backpropagation*, será projetada uma rede com *n* entradas, 1 camada intermediária contendo *k* unidades e *m* unidades de saída. Por questões de simplificação, a discussão a seguir considera apenas um único par de entrada e saída desejada, sendo a ideia empregada posteriormente generalizada para *p* padrões de treinamento.

O peso associado à conexão da entrada *i* até a unidade oculta *j* é denotado por $w_{i,j}^{(1)}$, assim como aquele associado à conexão da unidade oculta *j* até a unidade de saída l é denotado por $w_{j,l}^{(2)}$. O *bias*, $-\theta$, de cada nó é implementado como um peso adicional associado a uma entrada unitária, sendo denotado por $w_{n+1,j}^{(1)}$, quando localizado na conexão dessa entrada constante com uma unidade oculta *j* e denotado por $w_{k+1,l}^{(2)}$, quando localizado na conexão da entrada constante com uma unidade de saída l. A Figura A1.6 mostra a rede neural utilizada.

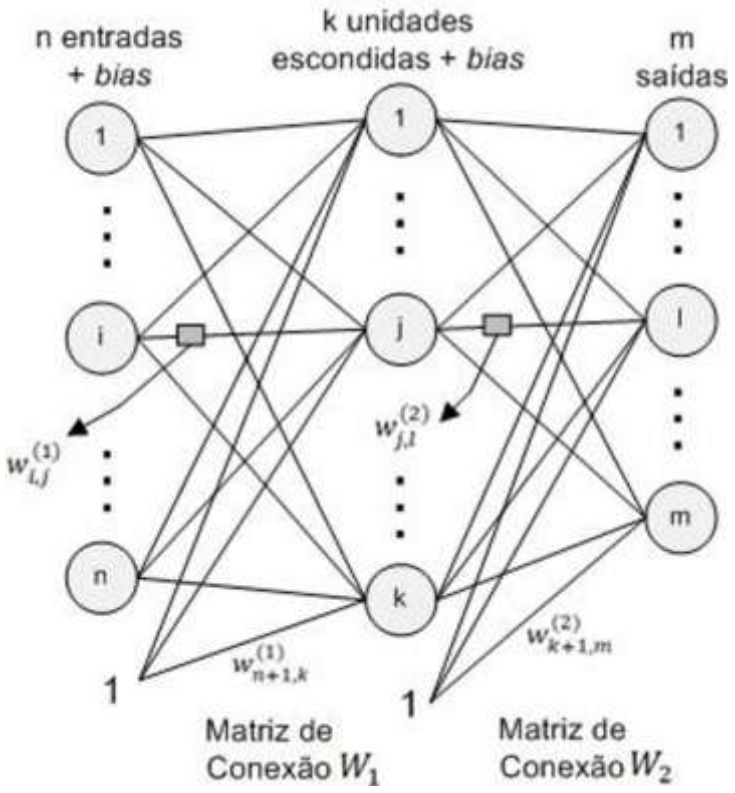


Figura A1.6 Rede neural de três camadas.

Denotando-se $\overline{W_1}$ como sendo a matriz da ordem $(n+1)\times k$ de pesos entre as entradas $\hat{o} = (o_1, \ldots, o_n, 1)$ e as unidades escondidas e $\overline{W_2}$, a matriz da ordem $(k+1)\times m$ de pesos entre as unidades escondidas e as de saída, têm-se que[29]:

$$\overline{W_1} = \begin{bmatrix} w_{1,1}^{(1)} & \cdots & w_{1,j}^{(1)} & \cdots & w_{1,k}^{(1)} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ w_{i,1}^{(1)} & \cdots & w_{i,j}^{(1)} & \cdots & w_{i,k}^{(1)} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ w_{n+1,1}^{(1)} & \cdots & w_{n+1,j}^{(1)} & \cdots & w_{n+1,k}^{(1)} \end{bmatrix} e\ \overline{W_2} = \begin{bmatrix} w_{1,1}^{(2)} & \cdots & w_{1,j}^{(2)} & \cdots & w_{1,m}^{(2)} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ w_{i,1}^{(2)} & \cdots & w_{i,j}^{(2)} & \cdots & w_{i,m}^{(2)} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ w_{k+1,1}^{(2)} & \cdots & w_{k+1,j}^{(2)} & \cdots & w_{k+1,m}^{(2)} \end{bmatrix} \quad \text{(A1.6)}$$

A excitação, $\mathrm{net_j}$, da j-ésima unidade oculta é dada por:

$$net_j = \sum_{i=1}^{n+1} w_{i,j}^{(1)} \hat{o}_i \quad \text{(A1.7)}$$

e, sabendo-se que sua função de ativação é uma sigmóide, sua saída é calculada como segue:

$$o_j^{(1)} = s(net_j) = s\left(\sum_{i=1}^{n+1} w_{i,j}^{(1)}\, \hat{o}_i\right) \quad \text{(A1.8)}$$

Considerando $\underline{\hat{o}}^{(1)} = \left(o_1^{(1)}, \ldots, o_k^{(1)}, 1\right)$ como sendo formado pelo vetor das saídas das unidades escondidas, juntamente com uma constante unitária, e $\underline{\hat{o}}^{(2)} = \left(o_1^{(2)}, \ldots, o_m^{(2)}\right)$, como sendo a saída da rede, pode-se escrever esses resultados sob a forma de representação de matrizes:

$$\underline{\hat{o}}^{(1)} = s\left(\underline{\hat{o}} \cdot \overline{W_1}\right) \text{ e} \quad \text{(A1.9)}$$

$$\underline{\hat{o}}^{(2)} = s\left(\underline{\hat{o}}^{(1)} \cdot \overline{W_2}\right) \quad \text{(A1.10)}$$

Essas expressões podem ser generalizadas para um número qualquer de camadas e permitem a computação direta do fluxo de informações na rede.

Levando em consideração os fatos anteriormente apresentados, pode-se analisar e discutir o funcionamento do algoritmo. A Figura A1.7 mostra parte da rede neural extendida para obtenção da função de erro. Sabendo-se que foi considerado apenas um par de padrão de entrada, o erro para *p* amostras pode ser calculado criando *p* redes como a mostrada a seguir, uma para cada par de treinamento, e adicionando as saídas de todas elas para produzir o erro total do conjunto de treino. Tal fato não será mostrado nas linhas posteriores.

[29] No Matlab®, as matrizes de pesos correspondem às transpostas de suas respectivas representações mostradas na Equação A1.6.

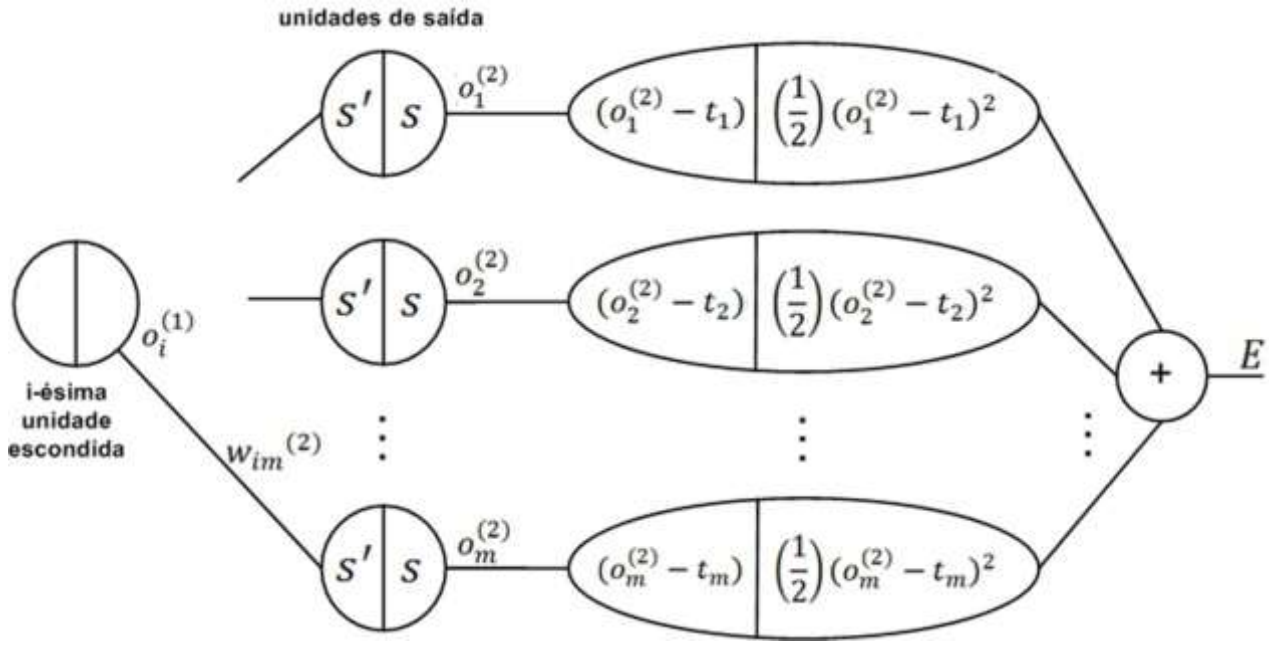


Figura A1.7 Esquemático empregado para computação do erro.

A rede é extendida com uma camada adicional de unidades computacionais, das quais seus lados direitos computam o desvio quadrático $\frac{1}{2}\left(o_i^{(2)} - t_i\right)^2$ para o i-ésimo componente do vetor de saída da rede e seus lados esquerdos armazenam a derivada $\left(o_i^{(2)} - t_i\right)$ desse desvio em relação ao mesmo componente. Após a inicialização randômica dos pesos da rede em cada conexão, o algoritmo *Backpropagation* é, então, usado para computar as correções necessárias, seguindo as etapas destacadas a seguir.

Primeira Etapa: Computação *feedforward*: o vetor de entrada é apresentado à rede e as respostas de cada camada, $\underline{\hat{o}}^{(1)}$ e $\underline{\hat{o}}^{(2)}$, são calculadas e armazenadas no lado direito de cada nó. As derivadas das funções primitivas são também armazenadas, porém no lado esquerdo nodal.

Segunda Etapa: Retropropagação do erro até a camada de saída: as derivadas armazenadas desde a saída da rede extendida até o nó *j* da camada de saída são multiplicadas e o valor acumulado até esse nó define o erro retropropagado (*backpropagated error*), $\delta_j^{(2)}$. O B-diagrama referente a essa etapa é mostrado na Figura A1.8.

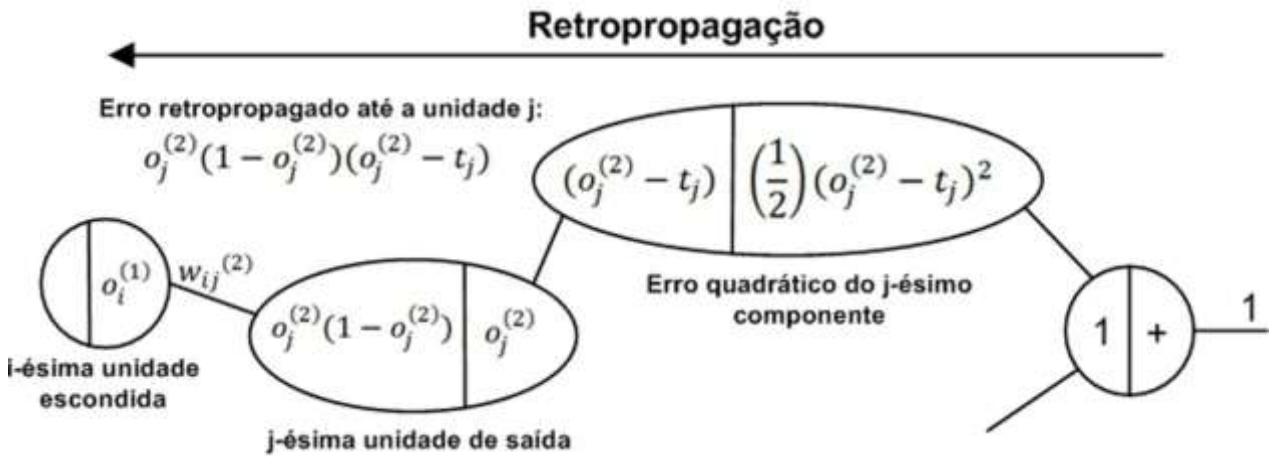


Figura A1.8 Retropropagação do erro até uma unidade de saída *j* qualquer.

Logo, tem-se que:

$$\delta_j^{(2)} = o_j^{(2)}\left(1 - o_j^{(2)}\right)\left(o_j^{(2)} - t_j\right) \tag{A1.11}$$

A derivada parcial do erro em relação ao peso localizado na conexão do nó escondido *i* ao nó de saída *j* pode ser calculada conforme a Regra da Cadeia, como segue:

$$\frac{\partial E}{\partial w_{ij}^{(2)}} = \frac{\partial E}{\partial o_j^{(2)}} \cdot \frac{\partial o_j^{(2)}}{\partial w_{ij}^{(2)}} \tag{A1.12}$$

Se $o_j^{(2)} = s\left(o_i^{(1)} \cdot w_{ij}^{(2)}\right)$, tem-se que:

$$\frac{\partial E}{\partial w_{ij}^{(2)}} = \left(o_j^{(2)} - t_j\right) \cdot \left(s' \cdot o_i^{(1)}\right) = \left[\left(o_j^{(2)} - t_j\right) \cdot o_j^{(2)} \cdot \left(1 - o_j^{(2)}\right)\right] \cdot o_i^{(1)} \tag{A1.13}$$

Portanto:

$$\frac{\partial E}{\partial w_{ij}^{(2)}} = \delta_j^{(2)} \cdot o_i^{(1)} \tag{A1.14}$$

Terceira Etapa: Retropropagação do erro até a camada escondida: cada unidade *j* na camada escondida está conectada a cada unidade q na camada de saída com um peso $w_{jq}^{(2)}$, para $q = 1, \ldots, m$. O erro retropropagado até a unidade *j* da camada escondida é calculado levando-se em consideração todos os possíveis caminhos da camada de saída a essa unidade, conforme Figura A1.9, e é expresso por:

$$\delta_j^{(1)} = o_j^{(1)} \cdot \left(1 - o_j^{(1)}\right) \cdot \sum_{q=1}^{m} w_{jq}^{(2)} \cdot \delta_q^{(2)} \tag{A1.15}$$

Portanto, analogamente à correspondente Equação A1.14, a derivada parcial do erro em relação ao peso localizado na conexão da entrada o_i ao nó escondido *j* é dada por:

$$\frac{\partial E}{\partial w_{ij}^{(1)}} = \delta_j^{(1)} \cdot o_i \qquad \text{(A1.16)}$$

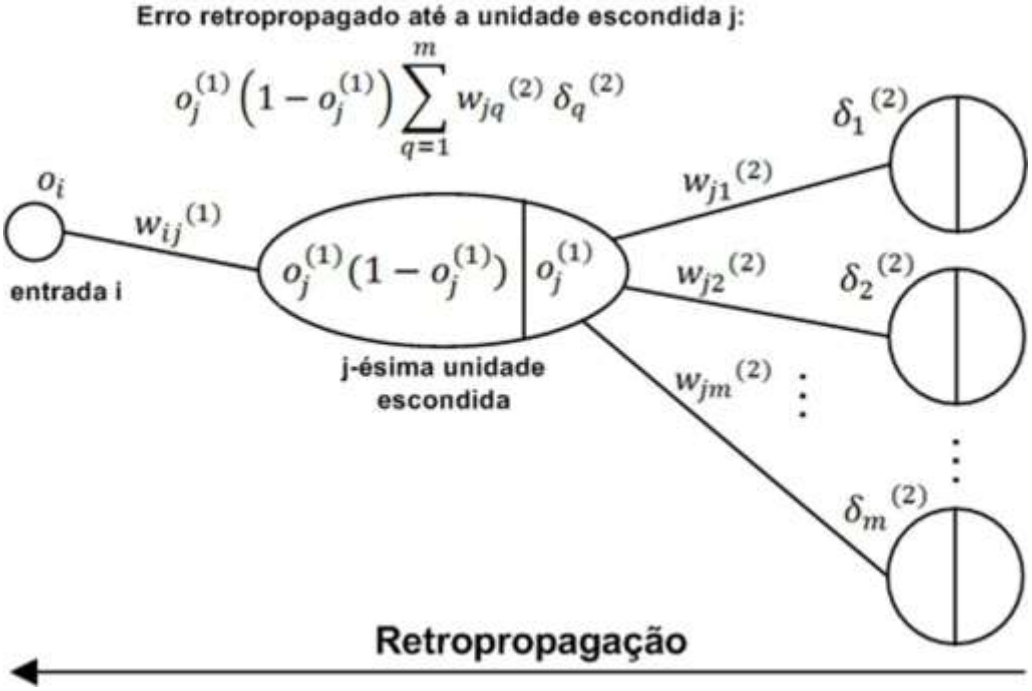


Figura A1.9 Retropropagação do erro até uma unidade escondida *j* qualquer.

Quarta Etapa: Atualização dos pesos: após a obtenção de todas as derivadas parciais, os pesos da rede são atualizados na direção negativa do gradiente, dada uma taxa de aprendizado γ, sendo somados aos incrementos:

$$\Delta w_{ij}^{(2)} = -\gamma \frac{\partial E}{\partial w_{ij}^{(2)}}, para\ i = 1, \dots, k+1;\ \ j = 1, \dots, m \quad \text{e} \qquad \text{(A1.18)}$$

$$\Delta w_{ij}^{(1)} = -\gamma \frac{\partial E}{\partial w_{ij}^{(1)}}, para\ i = 1, \dots, n+1;\ \ j = 1, \dots, k \qquad \text{(A1.19)}$$

Conforme discutido no Capítulo 3, essa atualização pode ser feita de modo incremental se os pesos são modificados sequencialmente após a apresentação de cada padrão de treinamento; ou de modo *batch*, quando a modificação ocorre após a apresentação de todos os padrões de treinamento, época a época. Nesse último caso, para o peso $w_{ij}^{(1)}$, por exemplo, todas suas correções, após *p* épocas, são dadas por:

$$\Delta_1 w_{ij}^{(1)}, \Delta_2 w_{ij}^{(1)}, \ldots, \Delta_p w_{ij}^{(1)} \tag{A1.20}$$

Portanto, a atualização do peso $w_{ij}^{(1)}$, na direção negativa do gradiente é feita com base no incremento total obtido a seguir:

$$\Delta w_{ij}^{(1)} = \Delta_1 w_{ij}^{(1)} + \Delta_2 w_{ij}^{(1)} + \cdots + \Delta_p w_{ij}^{(1)} \tag{A1.21}$$

A1.4 Fatores de Convergência do Algoritmo *Backpropagation*

Essa seção trata da questão da convergência do algoritmo *Backpropagation*, baseada em alguns fatores de aprendizado a serem discutidos a seguir, conforme Lim (1996).

a) Pesos iniciais: A inicialização dos pesos da rede neural afeta fortemente a solução final. Esses devem ser tipicamente inicializados com valores aleatórios pequenos; valores iguais dos pesos iniciais podem não treinar a rede adequadamente, caso a solução requeira pesos diferentes a serem desenvolvidos; já valores iniciais grandes podem levar à saturação da sigmóide e o sistema ficaria preso em um mínimo local. É aconselhável realizar por diversas vezes o treinamento de cada rede, assumindo valores aleatórios iniciais das matrizes de pesos, com o intuito de escapar de eventuais regiões de mínimos locais;

b) Taxa de aprendizado: Outro fator que afeta significantemente a convergência do *Backpropagation* é a taxa de aprendizado. Não há um único valor dessa taxa mais apropriado para diferentes casos de treinamento e seu valor é geralmente escolhido experimentalmente para cada problema. Um valor alto da taxa pode acelerar a convergência, mas pode resultar em *overshooting*, ou seja, sistema pode oscilar e se tornar instável, enquanto que um valor baixo tem efeito complementar, ou seja, a convergência irá demorar a acontecer, caso realmente aconteça. Além disso, outro problema verificado é que alguns valores atribuídos a essa taxa são melhores durante o início do treinamento, porém podem não ser tão bons ao final do mesmo. Portanto, uma metodologia mais eficiente a ser empregada consiste no o uso de uma taxa de aprendizado adaptativa. O intuito dessa abordagem consiste em checar se uma atualização de peso particular diminuiu o erro; em caso negativo, pode ter ocorrido *overshooting* e a taxa deve ser reduzida; por outro lado, em caso afirmativo, pode-se agir de forma conservativa e tentar aumentar a taxa;

c) Função de erro: A função quadrática do erro, vide Equação A1.1, não é a única escolha possível. O termo quadrático pode ser substituído por qualquer outra função diferenciável $F(o_i, t_i)$, a qual é minimizada quando seus argumentos são iguais;

d) Momento: Um método eficiente comumente empregado que permite uma alta taxa de aprendizado sem ocorrência de oscilações divergentes consiste na adição de um termo de momento ao gradiente descendente. Sua ideia é fornecer a cada peso alguma inércia ou momento de modo a evitar uma superfície de mínimo local, acelerando a convergência. Esse esquema é implementado fornecendo-se a cada peso uma contribuição da iteração anterior, conforme abaixo:

$$\Delta w(t) = -\gamma . \nabla E(t) + \alpha . \Delta w(t-1) \qquad \text{(A1.22)}$$

onde $\alpha \in [0,1]$ representa um parâmetro de momento e seu valor mais frequentemente usado é 0.9;

e) Regras de atualização: Apesar de o método do gradiente descendente ser uma técnica de otimização simples e eficiente, a teoria de otimização numérica pode ser aplicada para tornar a convergência do algoritmo *Backpropagation* mais rápida. Tal teoria forneceria um conjunto de técnicas rico e robusto a ser aplicado às redes neurais para melhorar as taxas de aprendizado;

f) Dados de treinamento e Generalização: Os dados de treinamento devem ser suficientes e adequados, ou seja, devem cobrir todo o espaço de entrada e, durante o processo de treinamento, padrões (pares de entrada e saída desejada) devem ser selecionados randomicamente do conjunto. Além disso, uma rede deve generalizar bem, ou seja, ser capaz de interpolar padrões de entrada novos à rede[30], de forma a evitar sua especialização ou *overfitting*, conforme discutido no Capítulo 3, e, ao mesmo tempo, evitar o aprendizado insuficiente neural ou *underfitting*, proveniente de um conjunto de dados pobre e não abrangente. É desejável alocar ao conjunto de treinamento as amostras que contiverem os valores mínimos e máximos relativos a quaisquer variáveis de entrada e saída da rede;

g) Número de unidades escondidas: O tamanho de uma camada escondida é uma questão fundamental em aplicações de redes do tipo *multilayer feedforward*. A análise dessa questão se torna difícil devido à complexidade do mapeamento neural e da natureza não-determinística de procedimentos de treinamento completados com sucesso.

[30] A fim de melhorar a habilidade de generalização da rede, é desejável que pequenas mudanças no espaço de entrada de padrões não modifiquem componentes da saída.

Usualmente, o número de nós em uma camada oculta é determinado experimentalmente, embora existam estudos recentes que têm comprovado que o tamanho de uma camada oculta pode ser determinado analiticamente. Pode-se estabelecer o tamanho de uma camada intermediária, baseando-se no desempenho do sistema como um todo. Para redes de tamanho razoável, o número de nós escondidos pode corresponder a uma pequena fração da camada de entrada. Se uma rede não consegue convergir a uma solução, um número maior de nós intermediários pode ser necessário, caso contrário, um número menor pode ser requerido.

Vale atentar ao fato de que o aumento do número de neurônios e de camadas da rede não implica sempre diretamente em melhoria de seu potencial de generalização. Considerando-se duas topologias que estão generalizando com o mesmo grau de precisão, é desejável selecionar aquela composta pela menor quantidade de neurônios, pois demonstra que ela foi capaz de extrair mais conhecimento, durante o processo de aprendizado.

Apêndice 2: Condições Singulares de Giro

A seguir é mostrado o arquivo contendo as condições singulares de giro anexadas à base de dados (devido à simetria, são mostrados somente os dados gerados a partir de $\theta_{2g} < 0$):

% theta2_giro	theta1_giro	gamma_giro
%----------	----------	----------
-54.000000	-31.607107	20.000000
-53.900000	-31.559947	20.000000
-53.800000	-31.512734	20.000000
-53.700000	-31.465469	20.000000
-53.600000	-31.418150	20.000000
-53.500000	-31.370779	20.000000
-53.400000	-31.323356	20.000000
-53.300000	-31.275879	19.000000
-53.200000	-31.228350	19.000000
-53.100000	-31.180769	19.000000
-53.000000	-31.133135	19.000000
-52.900000	-31.085448	19.000000
-52.800000	-31.037709	19.000000
-52.700000	-30.989918	19.000000
-52.600000	-30.942074	19.000000
-52.500000	-30.894178	19.000000
-52.400000	-30.846229	19.000000
-52.300000	-30.798228	19.000000
-52.200000	-30.750175	19.000000
-52.100000	-30.702069	19.000000
-52.000000	-30.653911	19.000000
-51.900000	-30.605701	19.000000
-51.800000	-30.557438	19.000000
-51.700000	-30.509124	19.000000
-51.600000	-30.460757	19.000000
-51.500000	-30.412338	19.000000
-51.400000	-30.363867	19.000000
-51.300000	-30.315344	19.000000
-51.200000	-30.266769	19.000000
-51.100000	-30.218142	19.000000
-51.000000	-30.169463	19.000000
-50.900000	-30.120732	19.000000
-50.800000	-30.071949	19.000000
-50.700000	-30.023114	19.000000
-50.600000	-29.974228	19.000000
-50.500000	-29.925289	19.000000
-50.400000	-29.876299	19.000000
-50.300000	-29.827257	19.000000
-50.200000	-29.778163	19.000000
-50.100000	-29.729018	18.000000
-50.000000	-29.679821	18.000000
-49.900000	-29.630572	18.000000
-49.800000	-29.581272	18.000000
-49.700000	-29.531920	18.000000
-49.600000	-29.482516	18.000000
-49.500000	-29.433061	18.000000
-49.400000	-29.383555	18.000000
-49.300000	-29.333997	18.000000
-49.200000	-29.284387	18.000000
-49.100000	-29.234727	18.000000
-49.000000	-29.185015	18.000000
-48.900000	-29.135251	18.000000
-48.800000	-29.085437	18.000000
-48.700000	-29.035571	18.000000
-48.600000	-28.985654	18.000000
-48.500000	-28.935686	18.000000
-48.400000	-28.885666	18.000000
-48.300000	-28.835596	18.000000
-48.200000	-28.785474	18.000000
-48.100000	-28.735302	18.000000
-48.000000	-28.685078	18.000000
-47.900000	-28.634804	18.000000
-47.800000	-28.584478	18.000000
-47.700000	-28.534102	18.000000
-47.600000	-28.483675	18.000000
-47.500000	-28.433197	18.000000
-47.400000	-28.382668	18.000000
-47.300000	-28.332089	18.000000
-47.200000	-28.281458	18.000000
-47.100000	-28.230778	17.000000
-47.000000	-28.180046	17.000000
-46.900000	-28.129264	17.000000
-46.800000	-28.078432	17.000000
-46.700000	-28.027548	17.000000
-46.600000	-27.976615	17.000000
-46.500000	-27.925631	17.000000
-46.400000	-27.874597	17.000000
-46.300000	-27.823512	17.000000
-46.200000	-27.772377	17.000000
-46.100000	-27.721192	17.000000
-46.000000	-27.669956	17.000000
-45.900000	-27.618670	17.000000
-45.800000	-27.567334	17.000000
-45.700000	-27.515949	17.000000
-45.600000	-27.464512	17.000000
-45.500000	-27.413026	17.000000
-45.400000	-27.361490	17.000000
-45.300000	-27.309904	17.000000
-45.200000	-27.258269	17.000000
-45.100000	-27.206583	17.000000
-45.000000	-27.154847	17.000000
-44.900000	-27.103062	17.000000
-44.800000	-27.051227	17.000000
-44.700000	-26.999342	17.000000
-44.600000	-26.947408	17.000000
-44.500000	-26.895424	17.000000
-44.400000	-26.843390	17.000000
-44.300000	-26.791307	17.000000
-44.200000	-26.739175	17.000000
-44.100000	-26.686993	16.000000
-44.000000	-26.634762	16.000000
-43.900000	-26.582481	16.000000
-43.800000	-26.530152	16.000000
-43.700000	-26.477772	16.000000
-43.600000	-26.425344	16.000000
-43.500000	-26.372867	16.000000
-43.400000	-26.320340	16.000000
-43.300000	-26.267765	16.000000
-43.200000	-26.215140	16.000000
-43.100000	-26.162467	16.000000
-43.000000	-26.109744	16.000000
-42.900000	-26.056973	16.000000
-42.800000	-26.004153	16.000000
-42.700000	-25.951284	16.000000
-42.600000	-25.898367	16.000000
-42.500000	-25.845401	16.000000
-42.400000	-25.792386	16.000000
-42.300000	-25.739322	16.000000
-42.200000	-25.686211	16.000000
-42.100000	-25.633050	16.000000
-42.000000	-25.579842	16.000000
-41.900000	-25.526584	16.000000

-41.800000	-25.473279	16.000000	-34.200000	-21.285064	13.000000
-41.700000	-25.419925	16.000000	-34.100000	-21.228216	13.000000
-41.600000	-25.366524	16.000000	-34.000000	-21.171325	13.000000
-41.500000	-25.313074	16.000000	-33.900000	-21.114391	13.000000
-41.400000	-25.259576	16.000000	-33.800000	-21.057413	13.000000
-41.300000	-25.206029	16.000000	-33.700000	-21.000392	13.000000
-41.200000	-25.152435	16.000000	-33.600000	-20.943328	13.000000
-41.100000	-25.098793	15.000000	-33.500000	-20.886221	13.000000
-41.000000	-25.045103	15.000000	-33.400000	-20.829072	13.000000
-40.900000	-24.991366	15.000000	-33.300000	-20.771879	13.000000
-40.800000	-24.937580	15.000000	-33.200000	-20.714643	13.000000
-40.700000	-24.883747	15.000000	-33.100000	-20.657365	13.000000
-40.600000	-24.829866	15.000000	-33.000000	-20.600044	13.000000
-40.500000	-24.775937	15.000000	-32.900000	-20.542681	13.000000
-40.400000	-24.721961	15.000000	-32.800000	-20.485275	13.000000
-40.300000	-24.667937	15.000000	-32.700000	-20.427827	13.000000
-40.200000	-24.613866	15.000000	-32.600000	-20.370337	12.000000
-40.100000	-24.559748	15.000000	-32.500000	-20.312804	12.000000
-40.000000	-24.505582	15.000000	-32.400000	-20.255230	12.000000
-39.900000	-24.451369	15.000000	-32.300000	-20.197613	12.000000
-39.800000	-24.397109	15.000000	-32.200000	-20.139954	12.000000
-39.700000	-24.342801	15.000000	-32.100000	-20.082253	12.000000
-39.600000	-24.288447	15.000000	-32.000000	-20.024511	12.000000
-39.500000	-24.234045	15.000000	-31.900000	-19.966727	12.000000
-39.400000	-24.179596	15.000000	-31.800000	-19.908901	12.000000
-39.300000	-24.125101	15.000000	-31.700000	-19.851033	12.000000
-39.200000	-24.070559	15.000000	-31.600000	-19.793124	12.000000
-39.100000	-24.015969	15.000000	-31.500000	-19.735174	12.000000
-39.000000	-23.961333	15.000000	-31.400000	-19.677183	12.000000
-38.900000	-23.906651	15.000000	-31.300000	-19.619150	12.000000
-38.800000	-23.851921	15.000000	-31.200000	-19.561076	12.000000
-38.700000	-23.797145	15.000000	-31.100000	-19.502961	12.000000
-38.600000	-23.742323	15.000000	-31.000000	-19.444805	12.000000
-38.500000	-23.687454	15.000000	-30.900000	-19.386608	12.000000
-38.400000	-23.632539	15.000000	-30.800000	-19.328370	12.000000
-38.300000	-23.577577	15.000000	-30.700000	-19.270092	12.000000
-38.200000	-23.522570	14.000000	-30.600000	-19.211773	12.000000
-38.100000	-23.467515	14.000000	-30.500000	-19.153413	12.000000
-38.000000	-23.412415	14.000000	-30.400000	-19.095014	12.000000
-37.900000	-23.357269	14.000000	-30.300000	-19.036573	12.000000
-37.800000	-23.302077	14.000000	-30.200000	-18.978093	12.000000
-37.700000	-23.246838	14.000000	-30.100000	-18.919572	12.000000
-37.600000	-23.191554	14.000000	-30.000000	-18.861011	12.000000
-37.500000	-23.136224	14.000000	-29.900000	-18.802410	11.000000
-37.400000	-23.080849	14.000000	-29.800000	-18.743769	11.000000
-37.300000	-23.025427	14.000000	-29.700000	-18.685089	11.000000
-37.200000	-22.969960	14.000000	-29.600000	-18.626368	11.000000
-37.100000	-22.914448	14.000000	-29.500000	-18.567608	11.000000
-37.000000	-22.858889	14.000000	-29.400000	-18.508809	11.000000
-36.900000	-22.803286	14.000000	-29.300000	-18.449970	11.000000
-36.800000	-22.747637	14.000000	-29.200000	-18.391091	11.000000
-36.700000	-22.691943	14.000000	-29.100000	-18.332174	11.000000
-36.600000	-22.636203	14.000000	-29.000000	-18.273217	11.000000
-36.500000	-22.580418	14.000000	-28.900000	-18.214221	11.000000
-36.400000	-22.524589	14.000000	-28.800000	-18.155186	11.000000
-36.300000	-22.468714	14.000000	-28.700000	-18.096112	11.000000
-36.200000	-22.412794	14.000000	-28.600000	-18.037000	11.000000
-36.100000	-22.356830	14.000000	-28.500000	-17.977848	11.000000
-36.000000	-22.300820	14.000000	-28.400000	-17.918658	11.000000
-35.900000	-22.244766	14.000000	-28.300000	-17.859430	11.000000
-35.800000	-22.188667	14.000000	-28.200000	-17.800163	11.000000
-35.700000	-22.132523	14.000000	-28.100000	-17.740858	11.000000
-35.600000	-22.076335	14.000000	-28.000000	-17.681514	11.000000
-35.500000	-22.020103	14.000000	-27.900000	-17.622133	11.000000
-35.400000	-21.963826	13.000000	-27.800000	-17.562713	11.000000
-35.300000	-21.907504	13.000000	-27.700000	-17.503255	11.000000
-35.200000	-21.851139	13.000000	-27.600000	-17.443760	11.000000
-35.100000	-21.794729	13.000000	-27.500000	-17.384226	11.000000
-35.000000	-21.738275	13.000000	-27.400000	-17.324655	11.000000
-34.900000	-21.681777	13.000000	-27.300000	-17.265047	11.000000
-34.800000	-21.625235	13.000000	-27.200000	-17.205401	11.000000
-34.700000	-21.568649	13.000000	-27.100000	-17.145718	10.000000
-34.600000	-21.512020	13.000000	-27.000000	-17.085997	10.000000
-34.500000	-21.455346	13.000000	-26.900000	-17.026239	10.000000
-34.400000	-21.398629	13.000000	-26.800000	-16.966445	10.000000
-34.300000	-21.341868	13.000000	-26.700000	-16.906613	10.000000

-26.600000	-16.846744	10.000000	-19.000000	-12.197059	7.000000
-26.500000	-16.786839	10.000000	-18.900000	-12.134681	7.000000
-26.400000	-16.726897	10.000000	-18.800000	-12.072276	7.000000
-26.300000	-16.666918	10.000000	-18.700000	-12.009843	7.000000
-26.200000	-16.606903	10.000000	-18.600000	-11.947382	7.000000
-26.100000	-16.546852	10.000000	-18.500000	-11.884894	7.000000
-26.000000	-16.486764	10.000000	-18.400000	-11.822378	7.000000
-25.900000	-16.426641	10.000000	-18.300000	-11.759835	7.000000
-25.800000	-16.366481	10.000000	-18.200000	-11.697265	7.000000
-25.700000	-16.306285	10.000000	-18.100000	-11.634668	7.000000
-25.600000	-16.246053	10.000000	-18.000000	-11.572044	7.000000
-25.500000	-16.185786	10.000000	-17.900000	-11.509393	7.000000
-25.400000	-16.125483	10.000000	-17.800000	-11.446715	7.000000
-25.300000	-16.065145	10.000000	-17.700000	-11.384011	7.000000
-25.200000	-16.004771	10.000000	-17.600000	-11.321280	7.000000
-25.100000	-15.944362	10.000000	-17.500000	-11.258523	7.000000
-25.000000	-15.883917	10.000000	-17.400000	-11.195740	7.000000
-24.900000	-15.823438	10.000000	-17.300000	-11.132931	7.000000
-24.800000	-15.762923	10.000000	-17.200000	-11.070096	7.000000
-24.700000	-15.702374	10.000000	-17.100000	-11.007235	7.000000
-24.600000	-15.641790	10.000000	-17.000000	-10.944349	7.000000
-24.500000	-15.581171	10.000000	-16.900000	-10.881437	7.000000
-24.400000	-15.520518	9.000000	-16.800000	-10.818500	7.000000
-24.300000	-15.459830	9.000000	-16.700000	-10.755537	7.000000
-24.200000	-15.399108	9.000000	-16.600000	-10.692550	6.000000
-24.100000	-15.338351	9.000000	-16.500000	-10.629537	6.000000
-24.000000	-15.277561	9.000000	-16.400000	-10.566500	6.000000
-23.900000	-15.216736	9.000000	-16.300000	-10.503438	6.000000
-23.800000	-15.155878	9.000000	-16.200000	-10.440351	6.000000
-23.700000	-15.094986	9.000000	-16.100000	-10.377240	6.000000
-23.600000	-15.034060	9.000000	-16.000000	-10.314105	6.000000
-23.500000	-14.973100	9.000000	-15.900000	-10.250945	6.000000
-23.400000	-14.912108	9.000000	-15.800000	-10.187762	6.000000
-23.300000	-14.851081	9.000000	-15.700000	-10.124554	6.000000
-23.200000	-14.790022	9.000000	-15.600000	-10.061323	6.000000
-23.100000	-14.728929	9.000000	-15.500000	-9.998068	6.000000
-23.000000	-14.667804	9.000000	-15.400000	-9.934790	6.000000
-22.900000	-14.606645	9.000000	-15.300000	-9.871489	6.000000
-22.800000	-14.545454	9.000000	-15.200000	-9.808164	6.000000
-22.700000	-14.484230	9.000000	-15.100000	-9.744816	6.000000
-22.600000	-14.422974	9.000000	-15.000000	-9.681445	6.000000
-22.500000	-14.361685	9.000000	-14.900000	-9.618052	6.000000
-22.400000	-14.300364	9.000000	-14.800000	-9.554636	6.000000
-22.300000	-14.239010	9.000000	-14.700000	-9.491197	6.000000
-22.200000	-14.177625	9.000000	-14.600000	-9.427736	6.000000
-22.100000	-14.116207	9.000000	-14.500000	-9.364253	6.000000
-22.000000	-14.054758	9.000000	-14.400000	-9.300747	6.000000
-21.900000	-13.993277	9.000000	-14.300000	-9.237220	6.000000
-21.800000	-13.931765	8.000000	-14.200000	-9.173671	6.000000
-21.700000	-13.870220	8.000000	-14.100000	-9.110100	6.000000
-21.600000	-13.808645	8.000000	-14.000000	-9.046508	5.000000
-21.500000	-13.747038	8.000000	-13.900000	-8.982894	5.000000
-21.400000	-13.685400	8.000000	-13.800000	-8.919259	5.000000
-21.300000	-13.623731	8.000000	-13.700000	-8.855603	5.000000
-21.200000	-13.562031	8.000000	-13.600000	-8.791926	5.000000
-21.100000	-13.500301	8.000000	-13.500000	-8.728228	5.000000
-21.000000	-13.438539	8.000000	-13.400000	-8.664509	5.000000
-20.900000	-13.376747	8.000000	-13.300000	-8.600770	5.000000
-20.800000	-13.314925	8.000000	-13.200000	-8.537010	5.000000
-20.700000	-13.253072	8.000000	-13.100000	-8.473231	5.000000
-20.600000	-13.191189	8.000000	-13.000000	-8.409431	5.000000
-20.500000	-13.129277	8.000000	-12.900000	-8.345611	5.000000
-20.400000	-13.067334	8.000000	-12.800000	-8.281771	5.000000
-20.300000	-13.005361	8.000000	-12.700000	-8.217911	5.000000
-20.200000	-12.943359	8.000000	-12.600000	-8.154032	5.000000
-20.100000	-12.881327	8.000000	-12.500000	-8.090134	5.000000
-20.000000	-12.819265	8.000000	-12.400000	-8.026216	5.000000
-19.900000	-12.757175	8.000000	-12.300000	-7.962280	5.000000
-19.800000	-12.695055	8.000000	-12.200000	-7.898324	5.000000
-19.700000	-12.632906	8.000000	-12.100000	-7.834349	5.000000
-19.600000	-12.570728	8.000000	-12.000000	-7.770356	5.000000
-19.500000	-12.508521	8.000000	-11.900000	-7.706344	5.000000
-19.400000	-12.446285	8.000000	-11.800000	-7.642314	5.000000
-19.300000	-12.384021	8.000000	-11.700000	-7.578266	5.000000
-19.200000	-12.321729	7.000000	-11.600000	-7.514199	5.000000
-19.100000	-12.259408	7.000000	-11.500000	-7.450115	5.000000

-11.400000	-7.386012	4.000000
-11.300000	-7.321892	4.000000
-11.200000	-7.257755	4.000000
-11.100000	-7.193600	4.000000
-11.000000	-7.129428	4.000000
-10.900000	-7.065239	4.000000
-10.800000	-7.001033	4.000000
-10.700000	-6.936810	4.000000
-10.600000	-6.872571	4.000000
-10.500000	-6.808315	4.000000
-10.400000	-6.744042	4.000000
-10.300000	-6.679754	4.000000
-10.200000	-6.615449	4.000000
-10.100000	-6.551129	4.000000
-10.000000	-6.486793	4.000000
-9.900000	-6.422441	4.000000
-9.800000	-6.358074	4.000000
-9.700000	-6.293692	4.000000
-9.600000	-6.229294	4.000000
-9.500000	-6.164881	4.000000
-9.400000	-6.100454	4.000000
-9.300000	-6.036012	4.000000
-9.200000	-5.971555	4.000000
-9.100000	-5.907084	4.000000
-9.000000	-5.842599	4.000000
-8.900000	-5.778100	4.000000
-8.800000	-5.713586	3.000000
-8.700000	-5.649059	3.000000
-8.600000	-5.584519	3.000000
-8.500000	-5.519964	3.000000
-8.400000	-5.455397	3.000000
-8.300000	-5.390816	3.000000
-8.200000	-5.326223	3.000000
-8.100000	-5.261616	3.000000
-8.000000	-5.196997	3.000000
-7.900000	-5.132365	3.000000
-7.800000	-5.067721	3.000000
-7.700000	-5.003064	3.000000
-7.600000	-4.938396	3.000000
-7.500000	-4.873715	3.000000
-7.400000	-4.809023	3.000000
-7.300000	-4.744319	3.000000
-7.200000	-4.679603	3.000000
-7.100000	-4.614876	3.000000
-7.000000	-4.550138	3.000000
-6.900000	-4.485389	3.000000
-6.800000	-4.420629	3.000000
-6.700000	-4.355859	3.000000
-6.600000	-4.291077	3.000000
-6.500000	-4.226286	3.000000
-6.400000	-4.161484	3.000000
-6.300000	-4.096672	2.000000
-6.200000	-4.031850	2.000000
-6.100000	-3.967018	2.000000
-6.000000	-3.902177	2.000000
-5.900000	-3.837326	2.000000
-5.800000	-3.772466	2.000000
-5.700000	-3.707597	2.000000
-5.600000	-3.642718	2.000000
-5.500000	-3.577831	2.000000
-5.400000	-3.512935	2.000000
-5.300000	-3.448031	2.000000
-5.200000	-3.383118	2.000000
-5.100000	-3.318197	2.000000
-5.000000	-3.253268	2.000000
-4.900000	-3.188331	2.000000
-4.800000	-3.123386	2.000000
-4.700000	-3.058433	2.000000
-4.600000	-2.993474	2.000000
-4.500000	-2.928506	2.000000
-4.400000	-2.863532	2.000000
-4.300000	-2.798551	2.000000
-4.200000	-2.733563	2.000000
-4.100000	-2.668568	2.000000
-4.000000	-2.603567	2.000000
-3.900000	-2.538559	2.000000
-3.800000	-2.473545	1.000000
-3.700000	-2.408526	1.000000
-3.600000	-2.343500	1.000000
-3.500000	-2.278469	1.000000
-3.400000	-2.213432	1.000000
-3.300000	-2.148389	1.000000
-3.200000	-2.083342	1.000000
-3.100000	-2.018289	1.000000
-3.000000	-1.953231	1.000000
-2.900000	-1.888169	1.000000
-2.800000	-1.823102	1.000000
-2.700000	-1.758031	1.000000
-2.600000	-1.692955	1.000000
-2.500000	-1.627875	1.000000
-2.400000	-1.562791	1.000000
-2.300000	-1.497704	1.000000
-2.200000	-1.432612	1.000000
-2.100000	-1.367518	1.000000
-2.000000	-1.302419	1.000000
-1.900000	-1.237318	1.000000
-1.800000	-1.172214	1.000000
-1.700000	-1.107107	1.000000
-1.600000	-1.041997	1.000000
-1.500000	-0.976884	1.000000
-1.400000	-0.911769	1.000000
-1.300000	-0.846652	1.000000
-1.200000	-0.781533	0.000000
-1.100000	-0.716412	0.000000
-1.000000	-0.651289	0.000000